

Lab 2-Initial Virtual Machines

Dawn M Inman

CSC-432 Computer and Network Security

Professor Nick Merante

March 27, 2020

### Abstract

This lab sets up the virtual network of the virtual machines that were previously set up and explored a bit in Lab 1. For this lab, CentOS 7 was renamed to router and configured into a router with both an external IP address for access to the internet, a firewall was put in place and configured for the NAT which allows traffic to come and go from the machines on the network, Kali Xfce was prepared for SSH access as well as having access to the internet. Proxmox was used only in the access it gives to use the QEMU interface to access both CentOS7 (names router) and the Kali VM. QEMU, however, was used extensively as the main access to both of these machines. A new access has been set up for the Kali VM that may be explored in the future, using the SSH to access the terminal via the router. Several issues were found when the wrong input for IP addresses happened twice, cause long delays for the student and major issues that needed help to go forward. Overall, this was a very successful lab with much accomplishment.

*Keywords:* CentOS 7, FirewallD, NAT, Kali Xfce, Penetration testing, Proxmox, KVM, QEMU, PuTTY.

### Lab 2-Initial Virtual Machines

Initializing virtual machines is necessary to be able to run well prepared, fully functioning virtual machines that will act like traditional dedicated machines. In order to make sure all things are working, with firewall intact while allowing passthrough of traffic, many steps need to be taken. This lab uses one virtual router and one virtual machine that are set up like a network would be set up with the router facing the external traffic, the firewall in place and the virtual machine behind the firewall and the router, creating an internal network. The virtual router being used is a CentOS 7 router which works with FirewallD and the virtual computer being run has a Kali Xfce. Both of these are also managed via Proxmox.

CentOS 7 is so named because it stands for Community ENTERprise Operating System. It is based on the Linux kernel, free and has been available since 2004. Red Hat Enterprise Linux is the origination of CentOS 7 so it is a compatible option when requiring Linux software. It is very popular with almost 30% of Linux web servers using it in 2011 and has been one of the most popular in hosting history. (CentOS Blog, 2020)

FirewallD uses zones and services to manage and control the traffic that goes to and from the system (network). It manages by using trust levels for interfaces and network connections. The zones and services take the place of iptables that were previously used, making it more user friendly. These can be configured to create control to and from flow of traffic, whether it will be allowed or disallowed according to trust level, according to “How to set up a firewall with FirewallD on CentOS7”. (November 11, 2019)

NAT stands for Network Address Translation. It allows an internal network (private network) to have one internet gateway. This gateway is the CentOS 7 router. The machines on

the internal network can have different IP addresses inside the network but when going outside of the router it will appear as if there is only one IP address being used. (Bischoff, 2019)

Kali Xfce is a newer Kali release. It is on the same line of Kali environments that have been created for Penetration Testing. There is a new feature called “Kali Undercover” which can make the display of Kali look like Windows 10. This can happen quickly so it is a type of stealth feature meant for blending in when in public areas. (Abrams, 2019) Other new features include KaliNetHunter KeX for Android which can install a full Kali desktop via Android, upgrading the kernel, Git powered documentation and adding PowerShell. (Elwood, 2019)

Proxmox is also being used by the systems but it is not used extensively in this lab. Proxmox VE hypervisor is based on GNU/Linux (Debian) and is open source. It has a central web-based management that does not require more installation. (Cheng, 2014) Version 5.4 is built specifically on Debian 9.8 with a “specially modified Linux Kernel 4.15”. (Proxmox, 2019) Proxmox is capable of two types of virtualization: OpenVZ and KVM. OpenVZ needs a patched Linux kernel so Linux guests are the only operating system type that can be created. In OpenVZ, the guests are called containers because they share the same architecture and kernel as the host operating system. (Cheng, 2014) KVM (Kernel-based Virtual Machine) is a modified Linux kernel built with the KVM module so that it can give hardware-assisted virtualization. Virtualization is performed by a software-based emulator (QEMU) which simulates the virtualized environment while KVM only exposes the /dev/kvm interface. (Cheng, 2014) “This converts Linux into a Type 1 (bare-metal) hypervisor.” (What is KVM?, 2020) Then QEMU or the software-based emulator will create the virtual machines on top of KVM. (What is KVM?, 2020) Proxmox VE is relatively simple to start working with but can be very in depth as Simon

M.C. Cheng has authored a book called Proxmox High Availability which goes into more detail when setting up a high availability virtual cluster. (Cheng, 2014)

PuTTY is an SSH client for Windows, Mac and Linux. It has a terminal window for access to the server used in this lab, the GNU/Linux server named chewy. (How to use PuTTY on Windows, 2020) SSH is a software package and means Secure Shell. It secures system administration and file transfers even though the networks are insecure. Tatu Ylonen is the inventor of SSH and OpenSSH which is an open source SSH program is based off of his free versions. (SSH(Secure Shell), 2020)

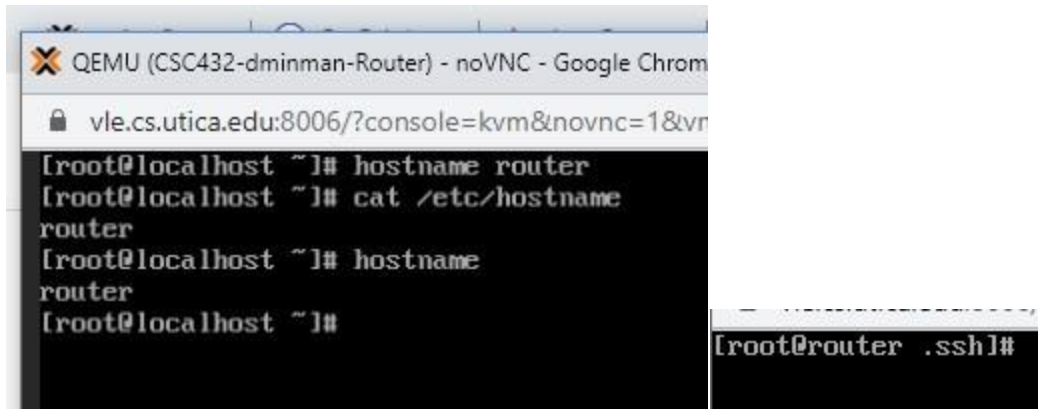
### **Objective**

This lab's purpose is to make the virtual server (Router) and virtual machine (VM) ready for labs that are to follow. First the router is renamed, then the external and internal IP addresses created with FirewallD being configured to support NAT. The Kali VM needed to be configured for networking on the internal network. Finally, SSH needs to be enabled to allow remote access to the Kali Machine with the SSH service set to run on boot.

The computer that is being used is a 2011 HP Pavillion dv7, i7 quad core processor and 16GB RAM with Windows 10 Pro operating system. Google Chrome is the internet browser being used for connecting to Proxmox including the Router and VM consoles. The VM then runs Firefox internet browser.

### **Results and Analysis<sup>1</sup>**

The first task is to change the name of the CentOS 7 server to router. The /etc/hostname file needed to be altered so that the name read router instead of localhost.localdomain. Once that was completed, hostname router was entered into the command line and the hostname command executed without an argument which is how the hostname was verified.

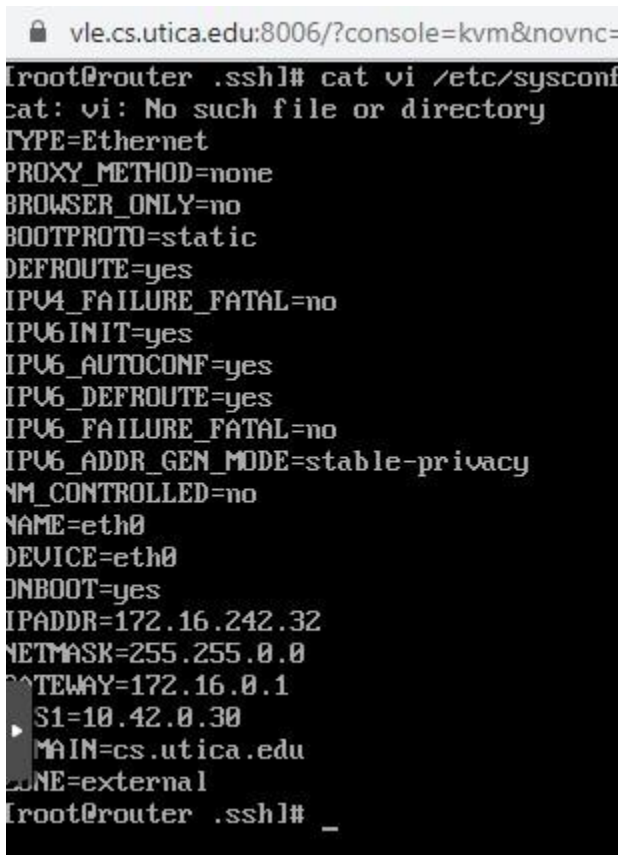


```
QEMU (CSC432-dminman-Router) - noVNC - Google Chrome
vle.cs.utica.edu:8006/?console=kvm&novnc=1&vnc=
[root@localhost ~]# hostname router
[root@localhost ~]# cat /etc/hostname
router
[root@localhost ~]# hostname
router
[root@localhost ~]#
[root@router .ssh]#
```

Figure 1 Server name changed to router

For the next step, the router has two internet addresses that need to be assigned to it so that it has both an external internet address that outside of the network will see, and inside that network where the network will connect to the router.

The external interface card is named eth0. The specs are as follows:

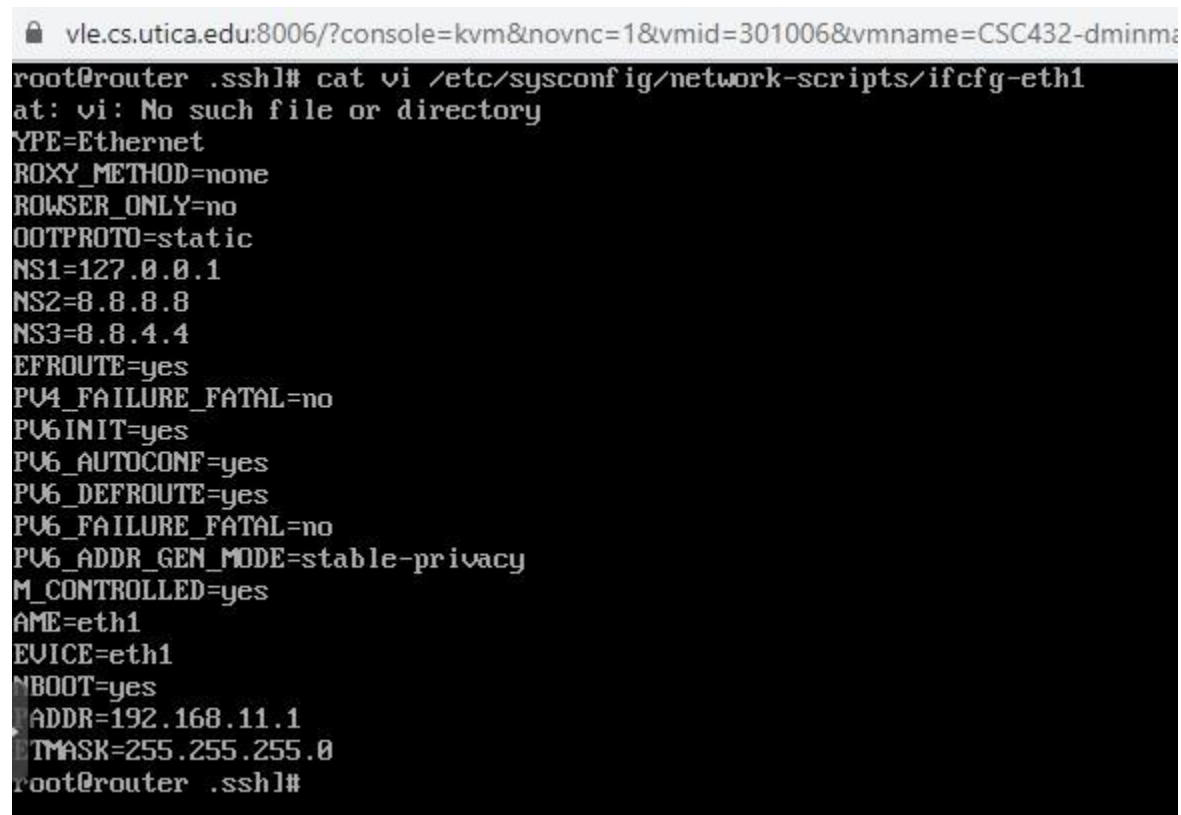


```
vle.cs.utica.edu:8006/?console=kvm&novnc=
[root@router .ssh]# cat vi /etc/sysconf
cat: vi: No such file or directory
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NM_CONTROLLED=no
NAME=eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=172.16.242.32
NETMASK=255.255.0.0
GATEWAY=172.16.0.1
DNS1=10.42.0.30
DOMAIN=cs.utica.edu
ZONE=external
[root@router .ssh]# _
```

Figure 2 eth0 specs

There were several items that needed to be changed or added. Those are bootproto=static, onboot=yes, ipaddr=172.16.242.32, gateway=172.16.0.1, dns1=10.42.0.30, domain=cs.utica.edu and zone=external. Capital letters and punctuation matter for the formatting so those were followed as necessary.

The internal interface then needed to be assigned so a similar process was completed but there were less requirements to format. The specs are as follows:

A screenshot of a terminal window with a black background and white text. The terminal shows a root user at a router in ssh mode. The user enters 'cat vi /etc/sysconfig/network-scripts/ifcfg-eth1'. The output shows various network configuration parameters for the eth1 interface, including TYPE, BOOTPROTO, IPADDR, NETMASK, and ONBOOT. The browser address bar at the top shows 'vle.cs.utica.edu:8006/?console=kvm&novnc=1&vmid=301006&vmname=CSC432-dminma'.

```
vle.cs.utica.edu:8006/?console=kvm&novnc=1&vmid=301006&vmname=CSC432-dminma
root@router .ssh# cat vi /etc/sysconfig/network-scripts/ifcfg-eth1
cat: vi: No such file or directory
TYPE=Ethernet
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.11.1
NETMASK=255.255.255.0
root@router .ssh#
```

Figure 3 eth1 specs

The details that needed to be modified or added were bootproto=static, onboot=yes, ipaddr=192.168.11.1 and netmask=255.255.255.0.

The system was restarted and everything worked perfectly so far. At this point, a mistake was made and the IP address for eth1 had been set at 192.168.11.10. This is the internet address for the VM, not the internal router address. Once that was pointed out by Professor Merante, it

was easily changed so that the connection could be made to the VM. This mistake was not caught until later in the lab, after the VM was set up and needed to connect to the internet to ping the google server. Having the correct IP address for each connection being made is the most important part of setting up a network. If all of the details are correct but one number is off, it will never connect. It is recommended to put together a drawing of the network that is to be set up. This helps show those IP addresses, how they relate and if anything is off or overlaps incorrectly. A drawing will be added later into the lab as an example of this.

For the next task, the firewall needs to be set up to support NAT. This file was created to allow IP forwarding:

```
vi /etc/sysctl.d/ip_forward.conf
```

Inside the file was written the text:

```
net.ipv4.ip_forward=1
```

Next the activating command was run

```
sysctl -p /etc/sysctl.d/ip_forward.conf
```

The return gives verification:

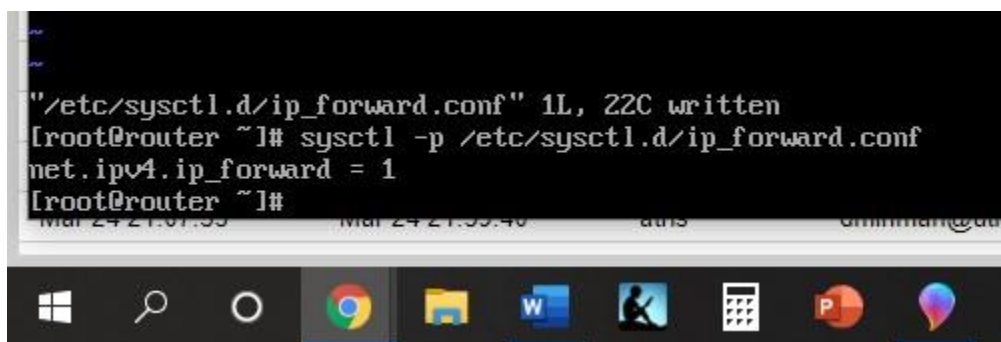


Figure 4 verification of forwarding

The router needs its IP addresses assigned to the firewall. This one assigns eth0:

```
Firewall-cmd --change-interface=eth0 --zone=external --permanent
```



The next two commands allow IP masquerading so the public cannot see the internal internet addresses, only the external one.

```
firewall-cmd --permanent --zone=external --add-masquerade
```

then a long command written all on one line:

```
firewall-cmd --permanent --direct --passthrough ipv4 -t nat -I POSTROUTING -o eth0 -j  
MASQUERADE -s 192.168.11.0/24
```

This long command gave a few issues as, in the other example, the wrong private net id and CIDR prefix was used at first. When the corrections were entered, it would say success, but when they were run as shown in the next two command steps, it would not work because it kept adding lines to the file, but that was not acceptable for what the program needs. It gave an error called an IP tables error. The error happened because of too many lines in the file.

```
[root@router ~]# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          172.16.0.1     0.0.0.0         UG      0      0      0 eth0
169.254.0.0      0.0.0.0        255.255.0.0     U        1002   0      0 eth0
169.254.0.0      0.0.0.0        255.255.0.0     U        1003   0      0 eth1
172.16.0.0       0.0.0.0        255.255.0.0     U        0      0      0 eth0
192.168.11.0     0.0.0.0        255.255.255.0   U        0      0      0 eth1
[root@router ~]#
```

Professor Merante pointed out that only one line can be inside the file, and that the file can be found at /etc/firewalld/direct.xml. When the incorrect lines were deleted, the next two commands ran smoothly and gave the correct output. This again shows that having the correct IP addresses for the network should be done ahead of time, not as the network is being built. Having a solid plan for the network IP addresses is always the best plan.

After the command that includes the netmask CIDR prefix, the default zone needs to be set to the internal network zone:

```
firewall-cmd --set-default-zone=internal
```

The firewall needs to be reloaded:

```
firewall-cmd --complete-reload
```

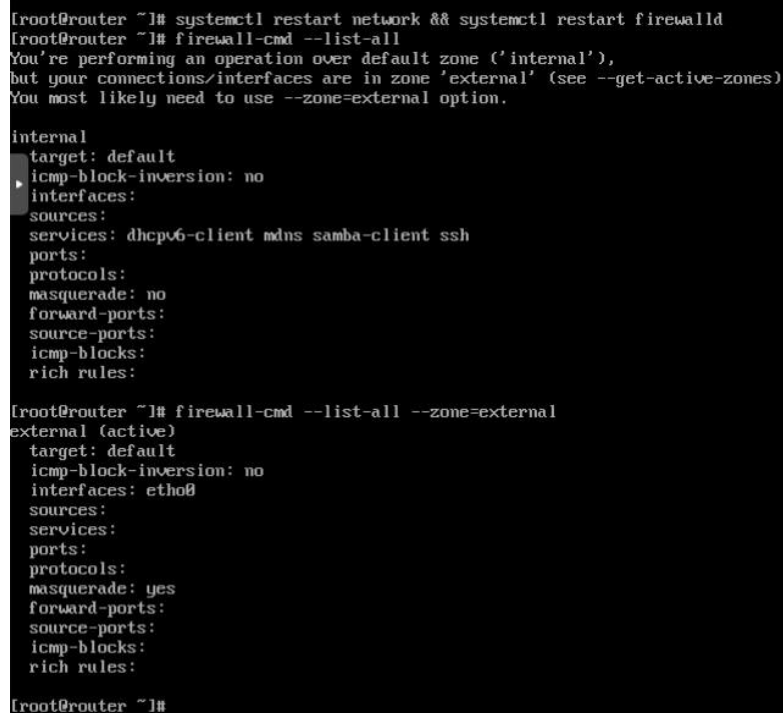
Then restart the network and the firewall services, both at the same time:

```
systemctl restart network && systemctl restart firewalld
```

The next commands prove that the firewall settings are permanent and persisted through reloading:

```
firewall-cmd --list-all
```

```
firewall-cmd --list-all --zone=external
```



```
[root@router ~]# systemctl restart network && systemctl restart firewalld
[root@router ~]# firewall-cmd --list-all
You're performing an operation over default zone ('internal'),
but your connections/interfaces are in zone 'external' (see --get-active-zones)
You most likely need to use --zone=external option.

internal
target: default
icmp-block-inversion: no
interfaces:
sources:
services: dhcpv6-client mdns samba-client ssh
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:

[root@router ~]# firewall-cmd --list-all --zone=external
external (active)
target: default
icmp-block-inversion: no
interfaces: etho8
sources:
services:
ports:
protocols:
masquerade: yes
forward-ports:
source-ports:
icmp-blocks:
rich rules:

[root@router ~]#
```

*Figure 5 firewall settings persist through reload*

At this point, the router can access the www. The proof is below for verification and it's IP addresses.

```

root@router .ssh1# ping www.google.com
PING www.google.com (172.217.10.68) 56(84) bytes of data.
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=1 ttl=55 time=23.3 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=2 ttl=55 time=23.1 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=3 ttl=55 time=23.2 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=4 ttl=55 time=23.3 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=5 ttl=55 time=23.2 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=6 ttl=55 time=22.8 ms
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=7 ttl=55 time=23.1 ms
^C
--- www.google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6008ms
rtt min/avg/max/mdev = 22.884/23.187/23.320/0.227 ms
root@router .ssh1# _

```

Figure 6 ping from [www.google.com](http://www.google.com)

```

root@router .ssh1# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b6:16:9e:db:1f:31 brd ff:ff:ff:ff:ff:ff
    inet 172.16.242.32/16 brd 172.16.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::b416:9eff:fedb:1f31/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 42:ea:db:54:2b:d9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.1/24 brd 192.168.11.255 scope global noprefixroute eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::1363:faa3:f9de:9088/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@router .ssh1# _

```

Figure 7 IP addresses

2.2. Next the Kali VM needed the correct IP address. The Network Manager tool made this easy, but the result took a long time to figure out as it was at this point when the incorrect IP address that was given to the internal interface on the router was found (eth1) via Professor Merante.

Right click on the internet button the top right side of the Kali display, then click edit connections.

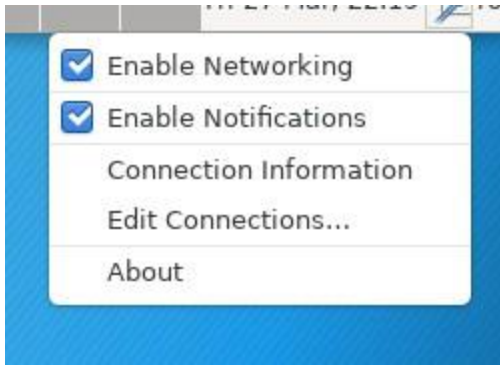


Figure 8 Right click for this drop-down menu for networks

Highlight the eth0 name and click the settings button at the bottom.

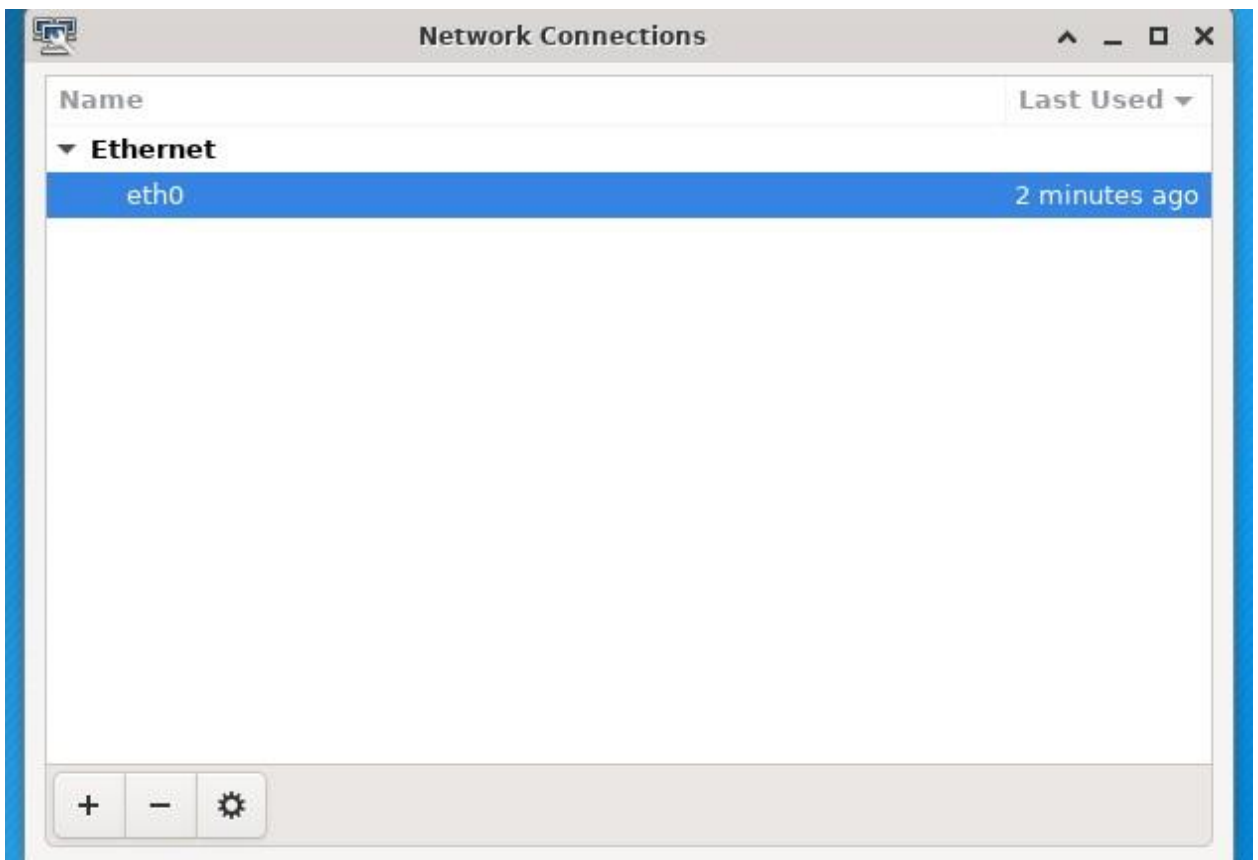
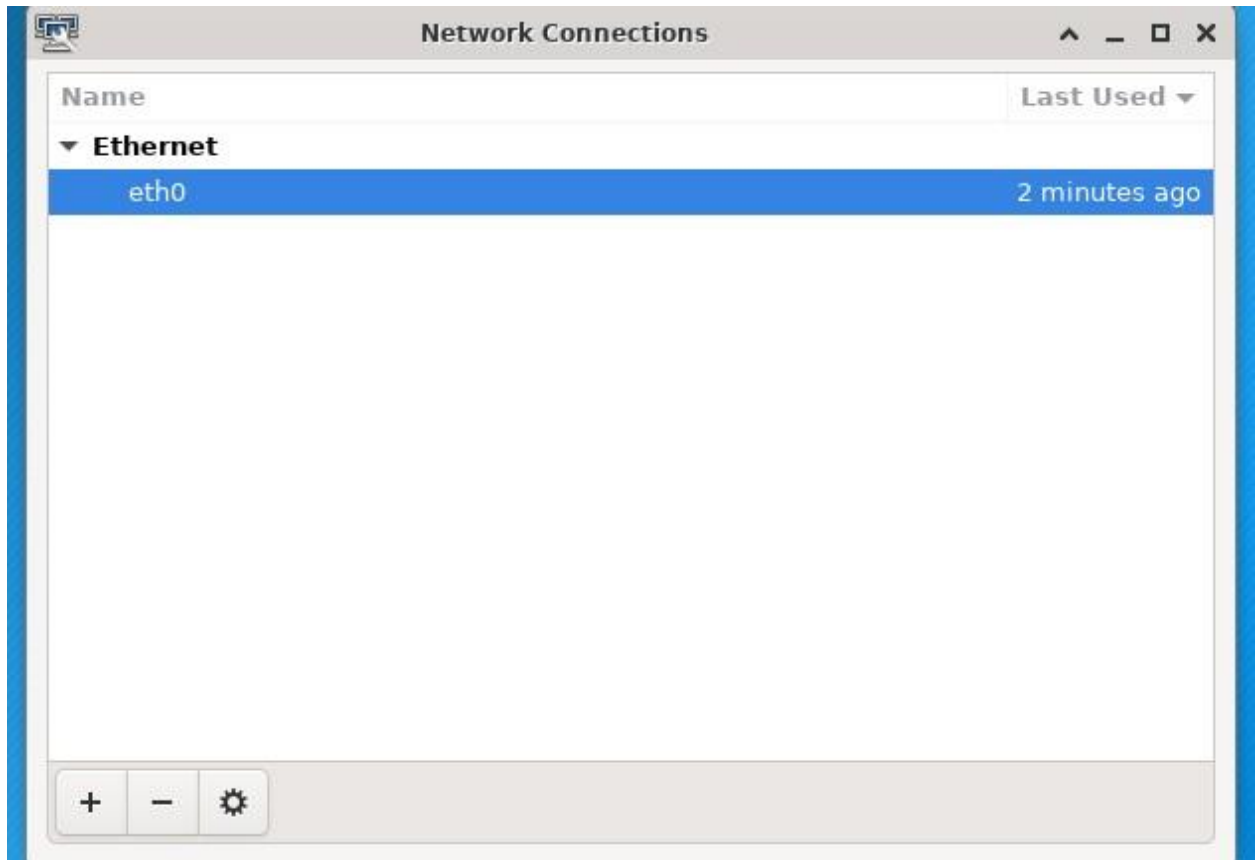


Figure 9 To get to eth0 settings

Next, changes need to be made to the settings.



*Figure 10 Network Manager settings*

The method needs to be changed from DHCP to Manual. The IP address of the Kali VM needs to be added, as well as the Netmask and Gateway. The DNS servers should also be added. In this example, there are three options listed for optimal connection options. When that is done, click save. The Kali VM should be connected to the internet.

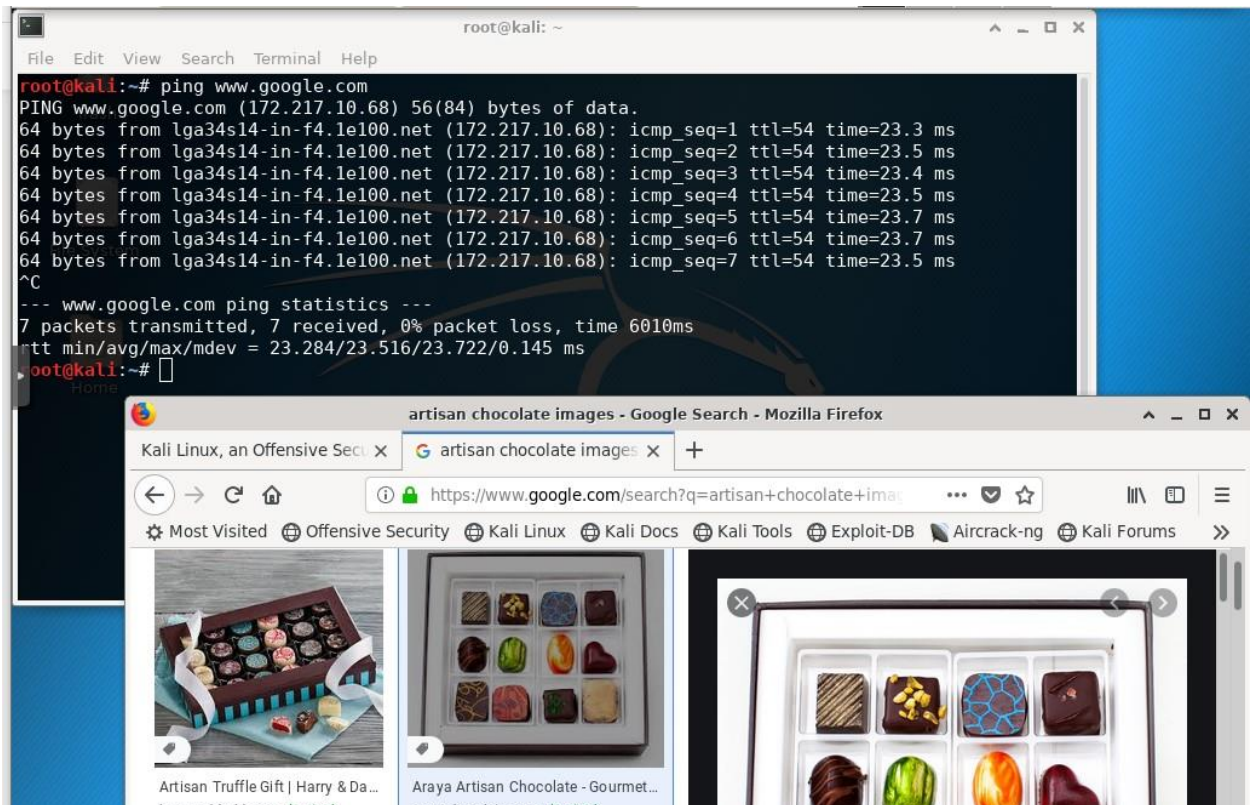


Figure 11 Kali Online

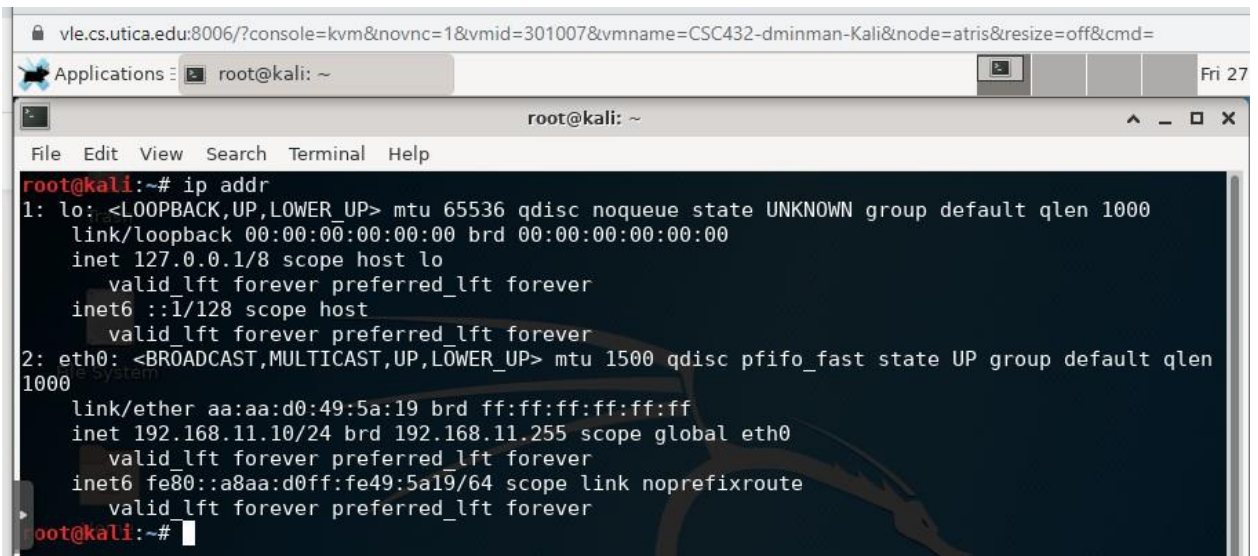


Figure 12 Kali ip addr

The last task was to enable SSH remote access on Kali. Editing of the file `/etc/ssh/sshd_config` was done so anyone with root access can login to it. The file was edited by changing

```
#PermitRootLogin prohibit-password
```

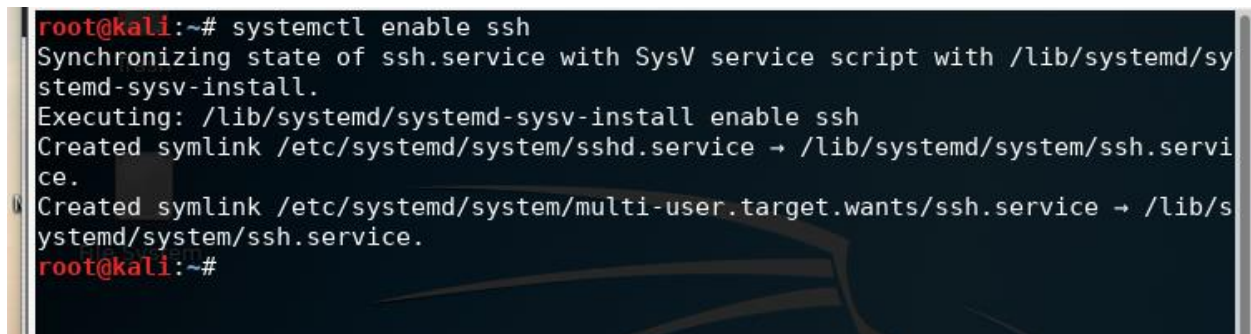
To

```
PermitRootLogin yes
```

Taking away the `#` in front makes the line go from being written as comment to a line of code or instruction to follow.

The SSH service was then set to run when boot up happens. This was accomplished by instructions via RoberRSeattle who said that he simply used

```
systemctl enable ssh
```

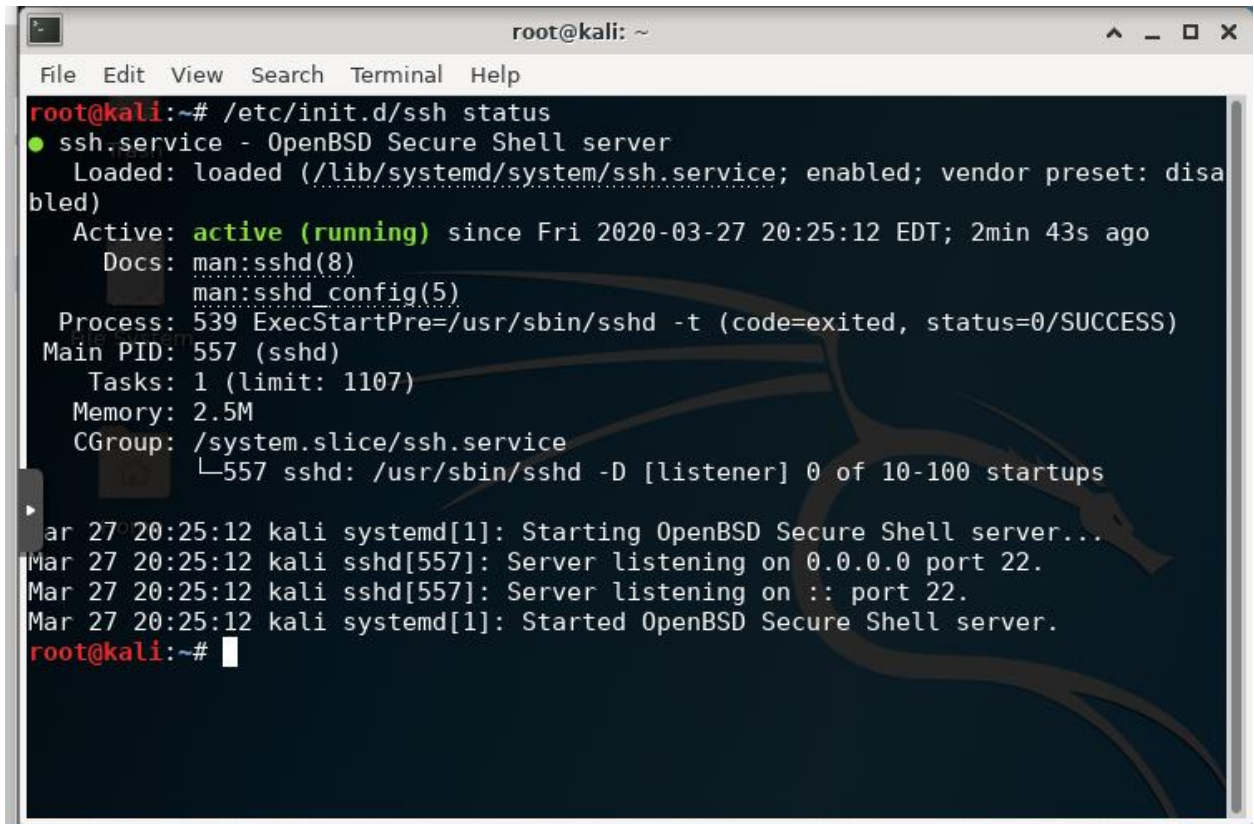


```
root@kali:~# systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/sy
stemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/sshd.service → /lib/systemd/system/ssh.servi
ce.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /lib/s
ystemd/system/ssh.service.
root@kali:~#
```

*Figure 13 SSH enable for run on boot*

This was very simple, yet, when Kali was rebooted, the SSH was on working properly without starting it on the terminal or using the PuTTY GUI.





```

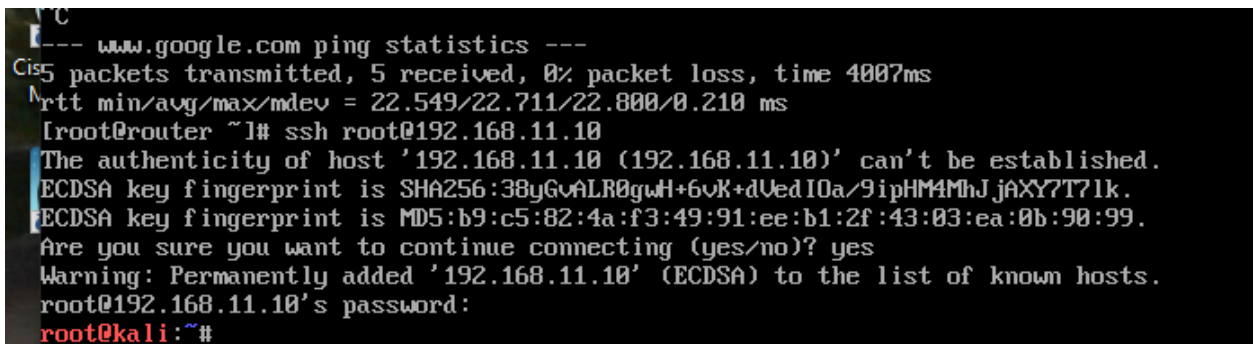
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# /etc/init.d/ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2020-03-27 20:25:12 EDT; 2min 43s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 539 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 557 (sshd)
    Tasks: 1 (limit: 1107)
   Memory: 2.5M
    CGroup: /system.slice/ssh.service
            └─557 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 27 20:25:12 kali systemd[1]: Starting OpenBSD Secure Shell server...
Mar 27 20:25:12 kali sshd[557]: Server listening on 0.0.0.0 port 22.
Mar 27 20:25:12 kali sshd[557]: Server listening on :: port 22.
Mar 27 20:25:12 kali systemd[1]: Started OpenBSD Secure Shell server.
root@kali:~#

```

Figure 14 SSH running on reboot

Also, access to the Kali Terminal was recorded via the SSH connection through the router (CentOS 7).



```

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 22.549/22.711/22.800/0.210 ms
[root@router ~]# ssh root@192.168.11.10
The authenticity of host '192.168.11.10 (192.168.11.10)' can't be established.
ECDSA key fingerprint is SHA256:38yGvALR0gwH+6vK+dUedIOa/9ipHM4MhJjAXY7T7lk.
ECDSA key fingerprint is MD5:b9:c5:82:4a:f3:49:91:ee:b1:2f:43:03:ea:0b:90:99.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.11.10' (ECDSA) to the list of known hosts.
root@192.168.11.10's password:
root@kali:~#

```

Figure 15 Control Terminal of Kali from the CentOS 7 router



The following is a very simple diagram of the current network configuration:

```

<-----external www internet-----External |router| IP address: 172.16.242.32
                                         |router|
                                         Internal |router| IP address: 192.168.11.1 ---to internal>
network----internal network connection -----Kali Machine IP address: 192.168.11.10

```

As shown, the router has two different IP addresses assigned to it so that the outside web side only sees that external internet address. This adds a layer of protection for the inside network, to be discussed more in following labs. Kali VM only have one internal IP address assigned to it as it is on the inside or private network.

### **Conclusion.**

This lab was successful at teaching how to set up a network with a router, a VM and a firewall. The student had never done this before and the learning involved was very great. Many mistakes were made, but each mistake brought more knowledge to the student. For example, it would seem like common sense that the Kali VM can not have the same IP address as the router Internal IP address, but the student needed to experience and see that to understand how the topology of the network is set up. Another area where the student learned was that, when making a mistake on an IP address or CIDR prefix, after it is entered wrong, it will need to be erased somewhere before the system will be able to read and understand the correct directions. If there are more than one set, it appears to the system as an error so it will not work properly. These directions have to be manually erased as the system does not erase them with additional input. The student really enjoyed the lab and hopes to make use of this with setting up a router at

home on a new desktop computer that was just purchased, but also at an internship and future job where learning to be precise will help the student excel long term. Excellent lab. Thank-you to the professor for his patience and help.

### References

- Abrams, Lawrence. (November 29, 2019). Kali Linux adds 'undercover' mode to impersonate Windows 10. BleepingComputer. Retrieved from <https://www.bleepingcomputer.com/news/security/kali-linux-adds-undercover-mode-to-impersonate-windows-10/>
- Bischoff, Paul. (March 28, 2019). What is a NAT firewall and how does it work? Comparitech. Retrieved from <https://www.comparitech.com/blog/vpn-privacy/nat-firewall/>
- Centos Blog (2020). What is Centos? Retrieved from <https://www.centosblog.com/what-is-centos/>
- Cheng, Simon M.C. (October 27, 2014). Basic concept of ProxMox Virtual Environment. Packt. Retrieved from <https://hub.packtpub.com/basic-concepts-proxmox-virtual-environment/>
- Elwood. (November 26, 2019). Kali Linux 2019.4 release. Kali Linux News. Retrieved from <https://www.kali.org/news/kali-linux-2019-4-release/>
- How to set up a firewall with FirewallD on CentOS7. (November 11, 2019). Linuxize. Retrieved from <https://linuxize.com/post/how-to-setup-a-firewall-with-firewalld-on-centos-7/>
- ProxMox. (April 11, 2019). ProxMox.com Retrieved from <https://www.proxmox.com/en/news/press-releases/proxmox-ve-5-4>
- RobetRSeattle. (March 13, 2017). Start ssh automatically on boot. AskUbuntu. Retrieved from <https://askubuntu.com/questions/892447/start-ssh-automatically-on-boot>
- SSH(Secure Shell). (2020). SSH.com. Retrieved from <https://www.ssh.com/ssh>