Homework 1 - Information Security (ICS344)

Alfaifi, Ammar

1 Installing VMs

In my case I installed two virtual machines. The first one is pre-built `vbox` of `Kali Linux` distro. The second VM, I chose install `Arch` Linux for its customiztion cababilities. The host operatin system is `MacBook Pro (13-inch, 2016, Two Thunderbolt 3 ports)` with `VirtualBox` as my VM manager.

2 Network Configuration

See Figure 1 for how I managed to change the netowrk settings for each VM.

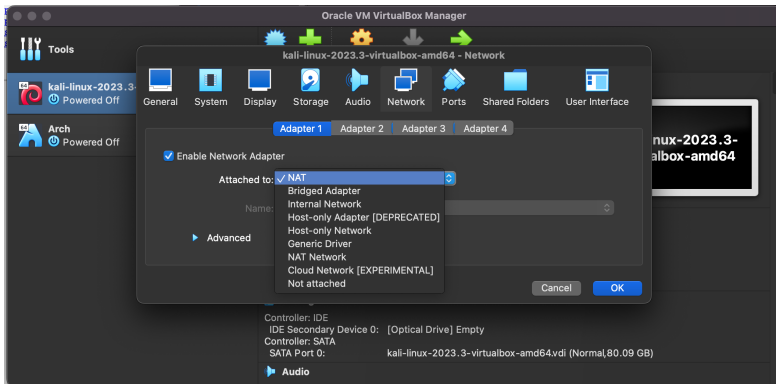

Figure 1: Changing each VM's networking mode.

2.1 Network Address Translation (NAT)

NAT networking allows the virtual machine to share the host's IP address and network connection. The host acts as a router, and the virtual machine is assigned an IP address in a private, isolated network. The host performs Network Address Translation, translating traffic between the VM and the external network. See Figure 2 for the VMs' IP addresses.

2.2 Bridged Networking

Bridged networking allows a virtual machine to appear as a separate device on the physical network to which your host computer is connected. This setting essentially bridges the virtual machine's network adapter with the host's network adapter, making it appear as if the virtual machine is directly connected to the physical network. See Figure 3; it appears that each VM it has its own IP address. As in Figure 1 we change to 'Bridge adapter', and in each VM's OS we set up the IP address and the gateway, as in Figure 4.

2.3 Host-Only Networking

Host-Only networking creates a private network that is isolated from both the external network and the host machine's network. VMs with host-only networking can communicate with each other and with the host but cannot access the internet or other external networks directly.

2.4 Ping each VM

As in Figure 6, I sent ping request form Kali to Arch, and vice versa. Also I did a ping from my hosting machine to the both of VM. All ping request had a response ICMP successfully.

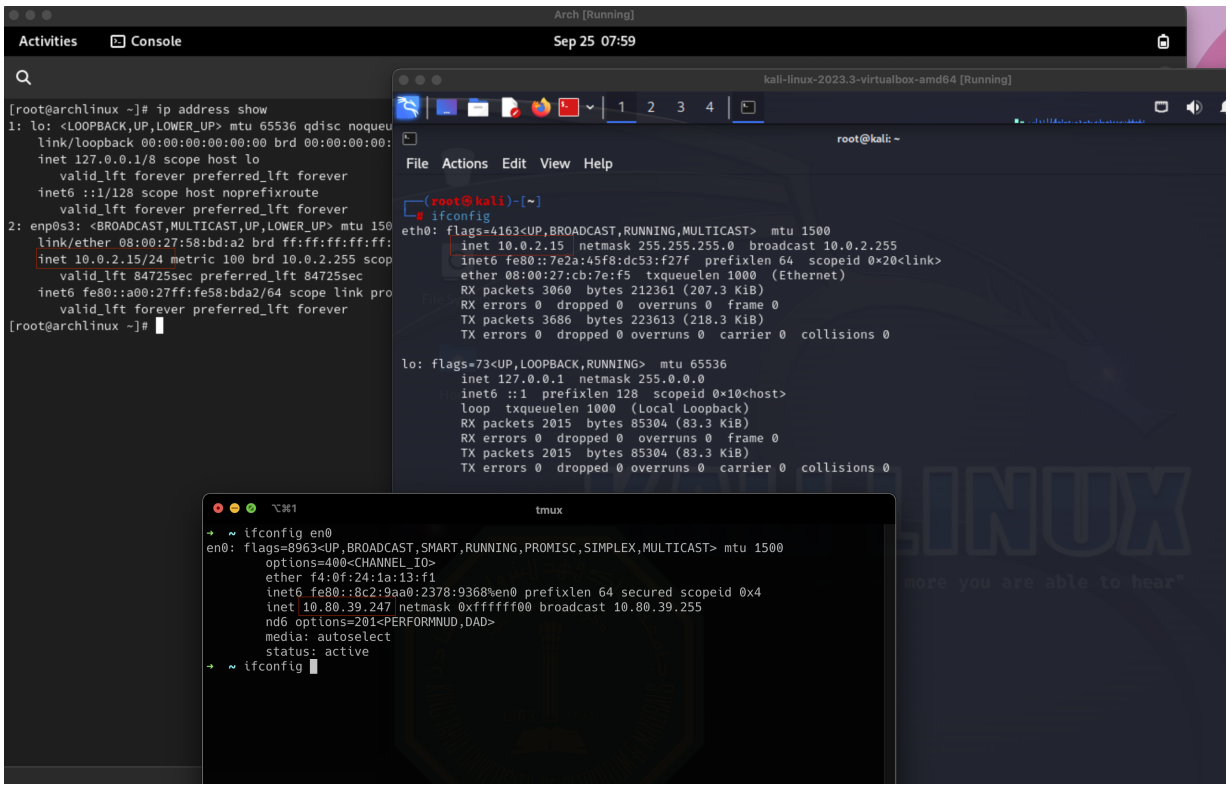

Figure 2: Left window is Arch VM, the right window is Kali VM, and the middle window is terminal from the host OS. You can see the two VMs share the same IP address, while the hosting machine has its unique IP address.

2.5 Accessing Internet

Here I use command `curl google.com` to do an HTTPS GET request to the domain `google.com`, As in Figure 7.

3 Network Security Tools

3.1 Using `tcpdump`

TCPdump: Command-line packet analyzer used for capturing and displaying network traffic.

3.1.1 Filter by IP

To use `tcpdump` and filter by IP address we can use the option `host`, which will filter in the any packet carries a source or destination matching the specifid address, see Figure 8

3.1.2 Getting password

I run a simple server on the local host, `http://localhost:8000`, which uses unsecure protocol, that is, HTTP (port 8000). Then I used `tcpdump` along with `grep` command to filter the packet that contains the 'user', as in Figure 9.

3.2 Wireshark

Graphical user interface (GUI) packet analyzer with powerful filtering and analysis capabilities. Offers a more comprehensive view of captured data with detailed protocol analysis and extensive filtering options.

3.2.1 Filter by IP

This is pretty easy in the GUI-based packets analyzer, as Wireshark. See Figure 10 how I did that.

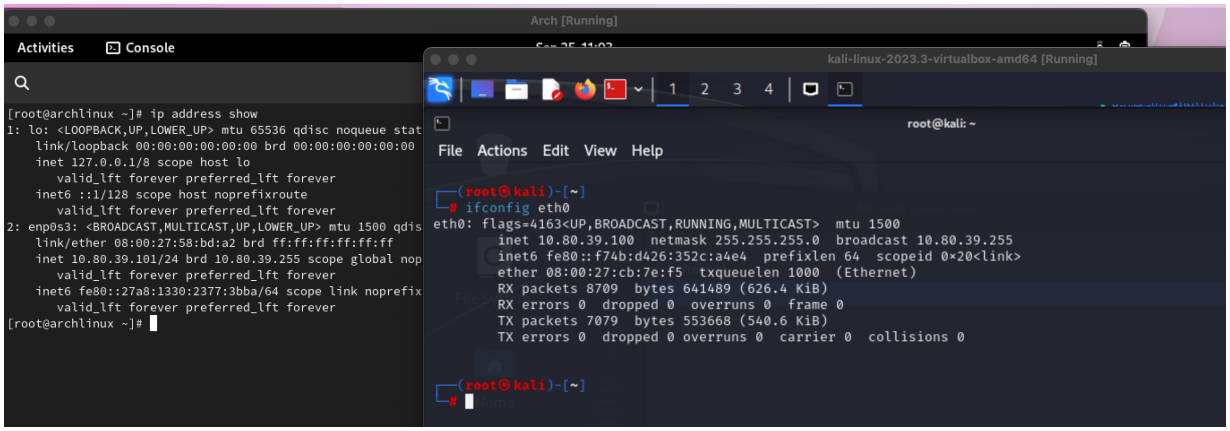

Figure 3: We see that for each machine it has its own unique IP address withing `10.80.93.0/24` network. As if it was directly connected to the hosting machine network.

3.2.2 Getting Password

It's similar to what I did in Figure 9, but I'll write the filter in the above filters box editing region, as in Figure 11.

3.3 Using `nmap`

Nmap, short for Network Mapper, is a versatile and widely used network scanning utility known for its multi-faceted capabilities. It serves network administrators, security professionals, and penetration testers by providing a comprehensive suite of features, including port scanning, OS detection, service version identification, and extensibility through the Nmap Scripting Engine (NSE).

3.3.1 Detecting OS of each VM

Using `nmap` along with the option `-O`, we can run this command on each IP address of each VM to detect the OS of each one, see Figure 12

3.3.2 Scan All Reserved Ports

To use `nmap` to scan all reserved ports we run the command `nmap -p 0-1023 IP_ADDRESS`, as I did in Figure 13

3.4 Using `amap`

Amap, or Application Mapper, focuses on a specialized niche within network scanning. Its primary mission is the precise identification of network services and applications running on open ports. Amap excels at this task by employing targeted probes and scrutinizing responses, making it especially adept at pinpointing less common or obscure services.

3.4.1 Detecting OS of each VM

`amap` is outdated, I could not find a binary for it nor it was bundled with Kali. I tried to build it form source but it requires old legacy libs so I gave up.

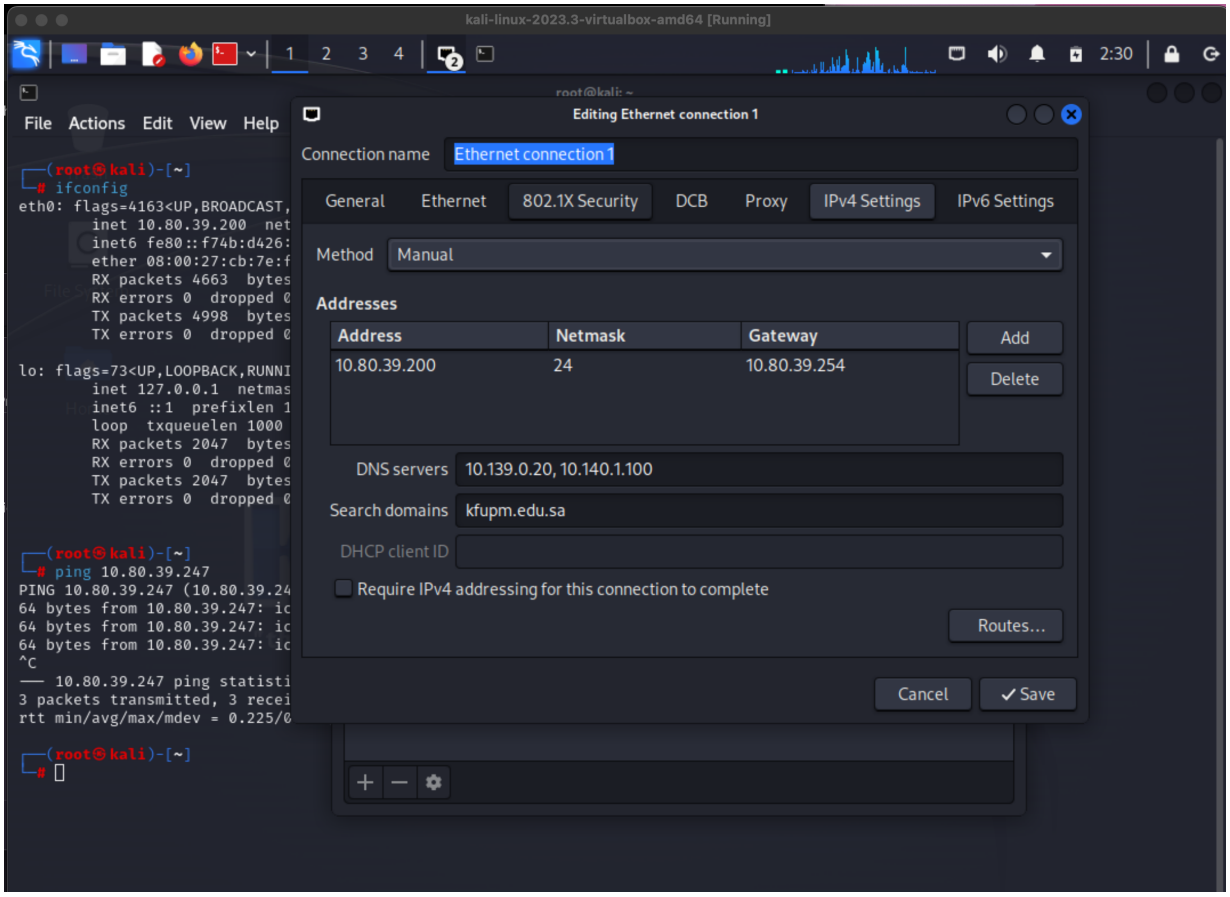

Figure 4: For Kali VM I set, IP address to `10.80.39.200`, gateway to `10.80.39.254`, and the subnet to `255.255.255.0` (or 24). And I copied and pasted the same DNS servers as in the hosting OS setting.

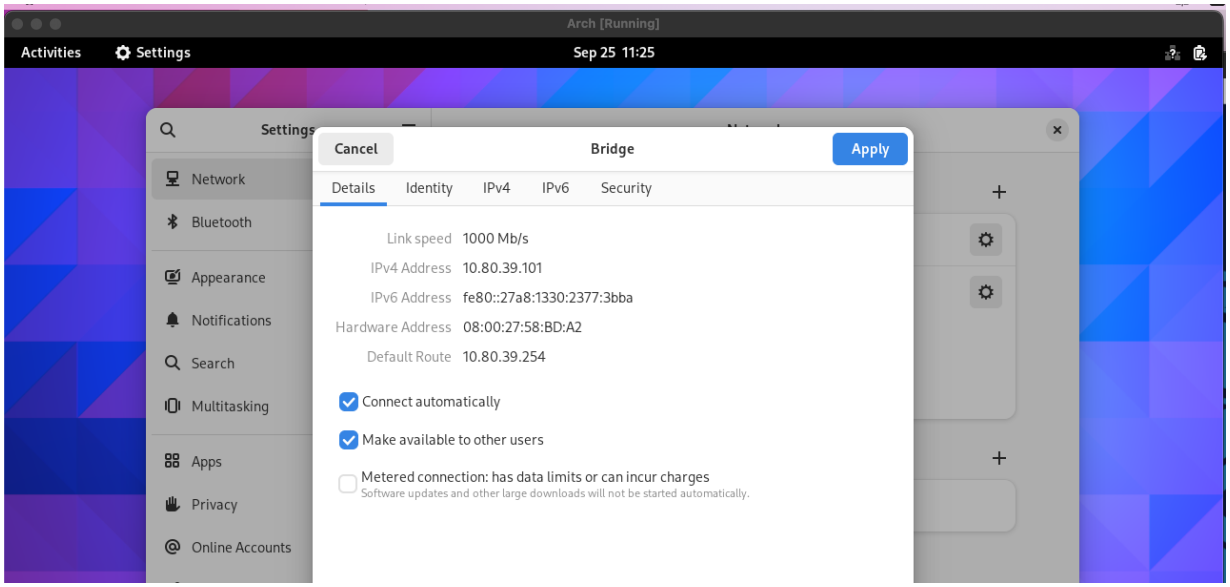

Figure 5: Same as in Figure 4 but for Arch VM but IP address is `10.80.39.101`

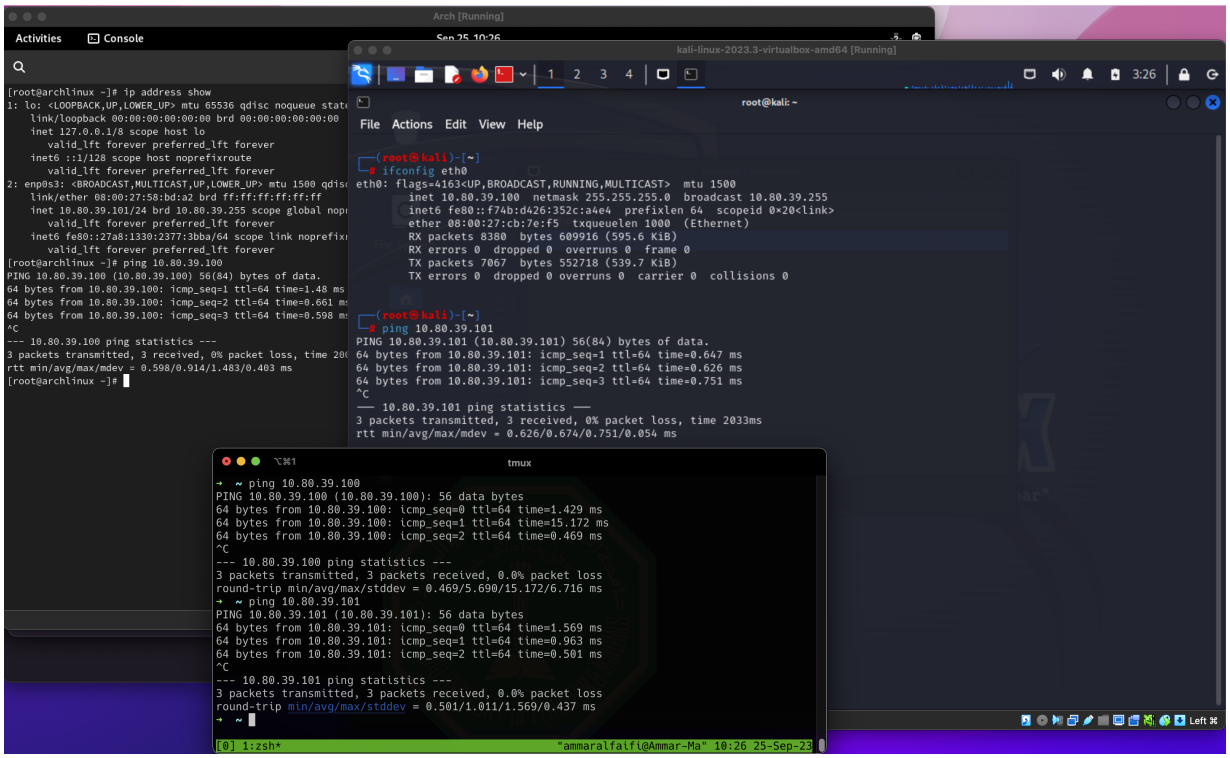

Figure 6: Left ping request from Arch to Kali, right is a ping request from Kali to Arch VM, and the middle window is ping request from hosting OS to both VMs.

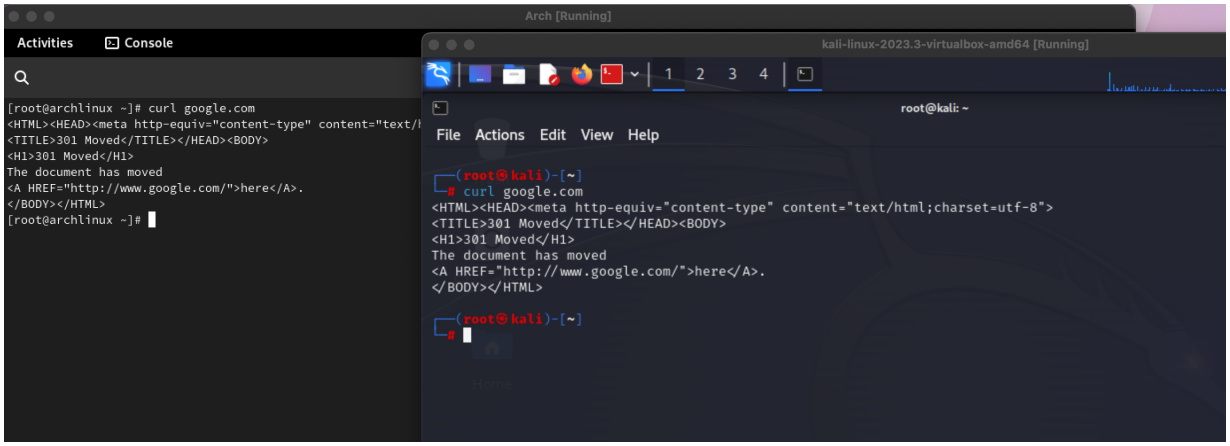

Figure 7: Accessing internet from each VM.

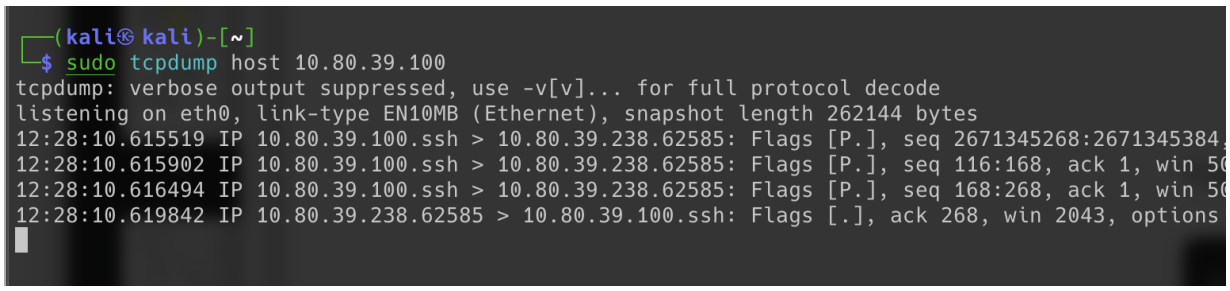

Figure 8: Using tcpdump with filtering by IP addresss, `10.80.39.100`, the Kali's IP address.

```
➜  ~ sudo tcpdump -A -i lo0 | grep -i 'secret'
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo0, link-type NULL (BSD loopback), snapshot length 524288 bytes
next=&csrfmiddlewaretoken=Cq4q9gZADE88TyUzlGUSkxX87Ao1tiGpgi1Bp9jtXXgSjd8USrACC72c0Qgu0vY5&username=secret&password=secret&login=
```

Figure 9: The username is 'secret' and the password is 'secret'

ip.src == 10.80.39.100

No.	Time	Source	Destination	Protocol	Length	Info
1561	78.097001	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=1/256, ttl=64 (no response found!)
1577	79.202602	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=2/512, ttl=64 (no response found!)
1590	80.266349	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=3/768, ttl=64 (no response found!)
1605	81.292985	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=4/1024, ttl=64 (no response found!)
1619	82.324572	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=5/1280, ttl=64 (no response found!)
1645	83.223959	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=6/1536, ttl=64 (no response found!)
1662	84.368791	10.80.39.100	10.80.39.238	ICMP	98	Echo (ping) request id=0xce8a, seq=7/1792, ttl=64 (no response found!)

Figure 10: We can filter IP address by writing in this format in the top text area box.

http

No.	Time	Source	Destination	Protocol	Length	Info
15	1.540628	127.0.0.1	127.0.0.1	HTTP	362	GET /getNotifications?ts=1695662117210&lrts=166
17	1.541507	127.0.0.1	127.0.0.1	HTTP/JSON	203	HTTP/1.1 200 OK , JavaScript Object Notation (a
27	1.544276	127.0.0.1	127.0.0.1	HTTP	326	GET /getNotifications HTTP/1.1
30	1.553138	127.0.0.1	127.0.0.1	HTTP/JSON	875	HTTP/1.1 200 OK , JavaScript Object Notation (a
61	6.541378	127.0.0.1	127.0.0.1	HTTP	362	GET /getNotifications?ts=1695662122219&lrts=166
63	6.542975	127.0.0.1	127.0.0.1	HTTP/JSON	203	HTTP/1.1 200 OK , JavaScript Object Notation (a
73	6.545333	127.0.0.1	127.0.0.1	HTTP	326	GET /getNotifications HTTP/1.1
76	6.549380	127.0.0.1	127.0.0.1	HTTP/JSON	875	HTTP/1.1 200 OK , JavaScript Object Notation (a
108	11.549493	127.0.0.1	127.0.0.1	HTTP	362	GET /getNotifications?ts=1695662127221&lrts=166
110	11.550561	127.0.0.1	127.0.0.1	HTTP/JSON	203	HTTP/1.1 200 OK , JavaScript Object Notation (a
120	11.553252	127.0.0.1	127.0.0.1	HTTP	326	GET /getNotifications HTTP/1.1
123	11.561592	127.0.0.1	127.0.0.1	HTTP/JSON	875	HTTP/1.1 200 OK , JavaScript Object Notation (a
156	16.537031	127.0.0.1	127.0.0.1	HTTP	1104	POST /account/login/ HTTP/1.1 (application/x-w
162	16.552253	127.0.0.1	127.0.0.1	HTTP	362	GET /getNotifications?ts=1695662132228&lrts=166

```
> Frame 156: 1104 bytes on wire (8832 bits), 1104 bytes captured (8832 bits) on interface lo0, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 64720, Dst Port: 8000, Seq: 1, Ack: 1, Len: 1048
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "next" = ""
  > Form item: "csrfmiddlewaretoken" = "vjS4OVepnD5uUUi4lI7XEtoon296sDnk9bPf4OyiHWdekzwpStNHW3tsgi1zZQF0"
  > Form item: "username" = "secret"
  > Form item: "password" = "secret"
```

Figure 11: Before launching Wireshark, I chose the Loopvack interface to capture packet from/to my lcoal server. The by using http filter I could expose the secret username and password.

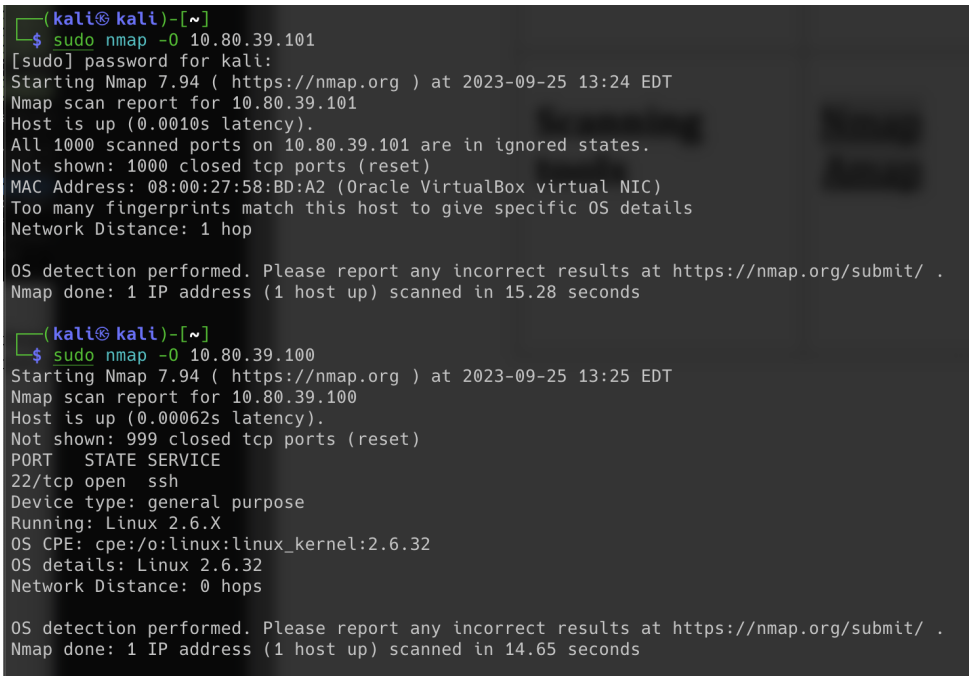

Figure 12: First run is to detect OS of Arch VM, the second run is to detect OS of the Kali VM.

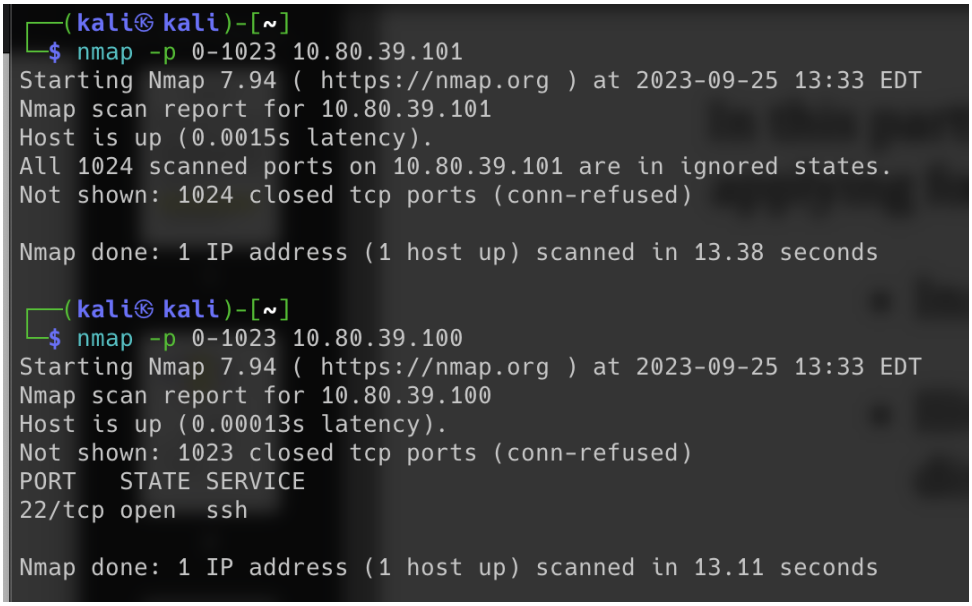

Figure 13: First run is to scan all reserved ports on Arch VM and the second run is for Kali VM. Note for Kali I run a `ssh` server to ease connecting to it, that's why it shows port 22 is awake.