

# **PONDICHERRY UNIVERSITY**

(A Central University)



**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE**

**Master of Computer Applications**

**NAME : MRITUNJAY KUMAR SOURAV**

**REG. NO. : 17352032**

**SEMESTER : 3<sup>rd</sup> YEAR, 5<sup>th</sup> SEMESTER**

**SUBJECT : MINI PROJECT REPORT**

**GUIDED BY : DR. R. SUNITHA**

# **NETWORK PACKET SNIFFER**

**By**

**MRITUNJAY KUMAR SOURAV**

**(Registration Number: 17352032)**

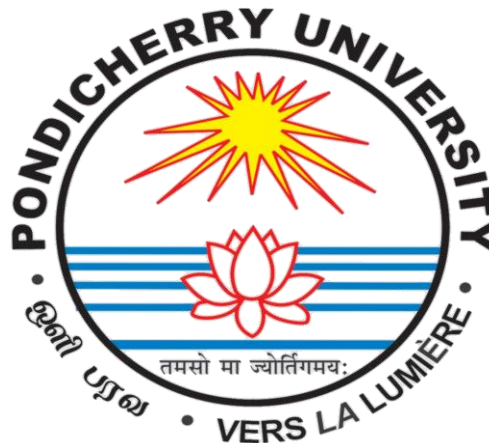
**Guided by**

**DR. R. SUNITHA**

**(Assistant professor. Department of Computer Science)**

**Mini Project report submitted in partial fulfillment of the requirements  
for the award of the degree of**

**MASTER OF COMPUTER APPLICATIONS**



**DEPARTMENT OF COMPUTER SCIENCE**

**SCHOOL OF ENGINEERING & TECHNOLOGY**

**PONDICHERY UNIVERSITY**

**NOVEMBER 2019**

## **BONAFIDE CERTIFICATE**

This is to certify that this project work entitled “**NETWORK PACKET SNIFFER**” is a bonafide record of work done by **Mr. MRITUNJAY KUMAR SOURAV** (Reg. Number 17352032) in the partial fulfillment for the degree of Master of Computer Applications of Pondicherry University.

This work has not been submitted elsewhere for the award of any other degree to the best of our knowledge.

### **INTERNAL GUIDE**

**Dr. R. Sunitha**

Assistant Professor,  
Department of Computer Science  
School of Engineering & Technology  
Pondicherry University  
Pondicherry – 605 014

### **HEAD OF THE DEPARTMENT**

**Dr. T. Chithralekha**

Professor and Head (i/c),  
Department of Computer Science  
School of Engineering & Technology  
Pondicherry University  
Pondicherry – 605 014

Submitted for the Viva-Voce Examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I am in great pleasure while I express my gratitude to all the people around me who have involved in this project and have helped me complete it successfully. This project has given me knowledge as well as confidence to work in a professional environment. Also, it has helped me to understand how to proceed through a complex process of learning and implementing the knowledge.

I would like to express my gratitude to the **Department of Computer Science, Pondicherry University** for offering this opportunity. It has made me realize my abilities and disabilities as well as how to overcome them.

I express my sincere gratitude to **Dr. R. Sunitha, Assistant Professor, Department of Computer Science, Pondicherry University**, who played the key role as my project guide. I thank her for mentoring me and strengthening my work with academic and mental support.

**(By- Mritunjay Kumar Sourav)**

# CONTENTS

## 1. Introduction

## 2. System Analysis

### 2.1 Existing System

### 2.2 Proposed System

### 2.3 System Specification

#### 2.3.1 Hardware Specification

#### 2.3.2 Software Specification

## 3 System Design

### 3.1 Module Design

### 3.2 Data Flow Diagram

## 4 Implementation, Configuration & Testing

## 5 Conclusion

## 6 Reference

## 7. Annexure

### 7.1 Coding

### 7.2 Interface

# **1. INTRODUCTION**

Packet sniffing is defined as a technique that is used to monitor every packet that crosses the network. A packet sniffer is a piece of hardware or software that monitors all network traffic. Using the information captured by the packet sniffers an administrator can identify erroneous packets and use the data to pinpoint bottlenecks and help to maintain efficient network data transmission . For most organizations packet sniffer is largely an internal threat. Packet sniffers can be operated in both switched and non switched environment. Determination of packet sniffing in a non switched environment is a technology that can be understood by everyone. In this technology all hosts are connected to a hub. There are a large number of commercial and non commercial tools are available that makes possible eavesdropping of network traffic. Now a problem comes that how this network traffic can be eavesdrop; this problem can be solved by setting network card into a special “promiscuous mode”. Now businesses are updating their network infrastructure, replacing aging hubs with new switches. The replacement of hub with new switches that makes switched environment is widely used because “it increases security”. However, the thinking behind is somewhat flawed. It cannot be said that packet sniffing is not possible in switched environment. It is also possible in switched environment.

## **HOW PACKET SNIFFER WORKS**

Packet sniffer’s working can be understood in both switched and non switched environment. For setup of a local network there exist machines. These machines have its own hardware address which differs from the other.

When a non switched environment is considered then all nodes are connected to a hub which broadcast network traffic to everyone. So as soon as a packet comes in the network, it gets transmitted to all the available hosts on that local network. Since all computers on that local network share the same wire, so in normal situation all machines

will be able to see the traffic passing through. When a packet goes to a host then firstly network card checks its MAC address, if MAC address matches with the host's MAC address then the host will be able to receive the content of that packet otherwise it will forward the packet to other host connected in the network. Now here a need arises to see the content of all packets that pass through the host. Thus we can say that when a host or machine's NIC is setup in promiscuous mode then all the packets that is designed for other machines, is captured easily by that host or machine.

When a switched environment is considered then all hosts are connected to a switch instead of a hub, it is called a switched Ethernet also. Since in switched environment packet sniffing is more complex in comparison to non switched network, because a switch does not broadcast network traffic. Switch works on unicast method, it does not broadcast network traffic, it sends the traffic directly to the destination host. This happens because switches have CAM Tables. These tables store information like MAC addresses, switch port and VLAN information. To understand working of packet sniffer in switched environment, an ARP cache table is considered. This is a table that stores both MAC addresses and IP addresses of the corresponding hosts. This table exists in local area network. Before sending traffic a source host should have its destination host, this destination host is checked in the ARP cache table. If destination host is available in the ARP cache then traffic will be sent to it through a switch, but if it is not available in the ARP cache then source host sends a ARP request and this request is broadcasted to all the hosts. When the host replies the traffic can be sent to it. This traffic is sent in two parts to the destination host. First of all it goes from the source host to the switch and then switch transfers it directly on the destination host. So sniffing is not possible.

## **SNIFFING METHODS USED**

1. IP based sniffing is the most commonly used method of packet sniffing. In this method a requirement of setting network card into promiscuous mode exist. When network card is set into promiscuous mode then host will be able to sniff all packets. A key point in the IP based sniffing is that it uses an IP based filter, and the packets matching the IP address filter is captured only. Normally the IP address filter is not set so it can capture all the packets. This method only works in non switched network managers to put security measures, such as firewalls, in place to prevent intrusion to the network.
2. MAC based sniffing is another method of packet sniffing. This is as like IP based sniffing. Same concept of IP based sniffing is also used here besides using an IP based filter. Here also a requirement of setting network card into promiscuous mode exists. Here in place of IP address filter a MAC address filter is used and sniffing all packets matching the MAC addresses.
3. ARP based Sniffing method works a little different. It does not put the network card into promiscuous mode. This is not necessary because ARP packets will be sent to us. This is an effective method for sniffing in switched environment. Here sniffing is possible due to of being stateless nature of Address Resolution Protocol.



## **PROJECT OBJECTIVE**

The objective of my project is to intercept and log traffic that passes over a digital network or part of a network.

## **USES OF PACKET SNIFFER**

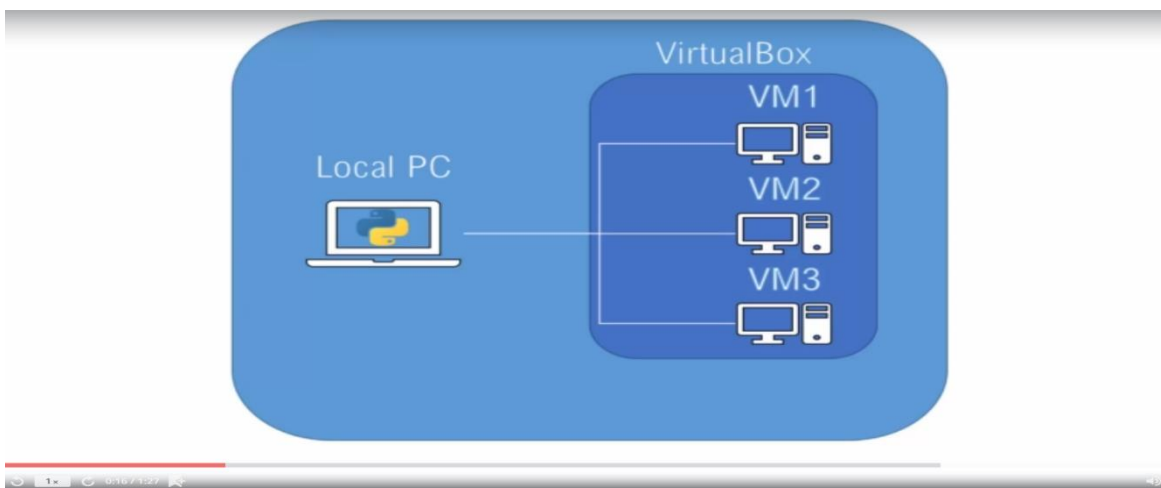
### **Packet sniffers can:**

- Analyze network problems
- Detect network intrusion attempts
- Detect network misuse by internal and external users
- Documenting regulatory compliance through logging all perimeter and endpoint traffic
- Gain information for effecting a network intrusion
- Isolate exploited systems
- Monitor WAN bandwidth utilization
- Monitor network usage (including internal and external users and systems)
- Monitor data in transit
- Monitor WAN and endpoint security status
- Gather and report network statistics
- Identify suspect content in network traffic
- Serve as primary data source for day-to-day network monitoring and management
- Spy on other network users and collect sensitive information such as login details or users cookies (depending on any content encryption methods that may be in use)
- Reverse engineer proprietary protocols used over the network

- Debug client/server communications
- Debug network protocol implementations
- Verify adds, moves and changes
- Verify internal control system effectiveness (firewalls, access control, Web filter, spam filter, proxy)

Packet capture can be used to fulfill a warrant from a law enforcement agency to wiretap all network traffic generated by an individual. Internet service providers and VoIP providers in the United States must comply with Communications Assistance for Law Enforcement Act regulations. Using packet capture and storage, telecommunications carriers can provide the legally required secure and separate access to targeted network traffic and are able to use the same device for internal security purposes. Collecting data from a carrier system without a warrant is illegal due to laws about interception. By using end-to-end encryption, communications can be kept confidential from telecommunication carriers and legal authorities.

## **Network Architecture**



## **PRACTICAL APPROACH**

- ✓ A practical approach of this title is developed by us in which we have shown actual packet capturing. This approach is mostly developed for:
- ✓ To make data identity stealing available by tracing the packets from the network.
- ✓ To provide an easy and effective way of sniffing of data packets.
- ✓ To provide a user friendly environment.
- ✓ It is possible only when the server code is running.

## **2. SYSTEM ANALYSIS**

### **2.1 EXISTING SYSTEM**

Existing system supports only the packet capturing there is no sniffing concept. It can show only the captured packet in the network and it can show only the size of the packet. In this application it cannot show the source machine and destination machine which are involved in the packet transferring.

### **2.1 PROPOSED SYSTEM**

In this application it can show the “packet sniffing” concept. In this manner it can show the captured packets and size of the packet and source and destination machine IP addresses which are involved in the packet transferring. It can show the working of different layers in graphical manner. It can give the complete information about the captured packet like which layers are involved and which protocols are involved at that time. And you have a facility to store the information of the packets. For developing this application we have made five modules they are as:

- ✓ Packet sniffing module
- ✓ Analyze layer module
- ✓ Analyze protocol module

## **2.3 SYSTEM SPECIFICATION**

### **2.3.1 HARDWARE SPECIFICATION**

- RAM : 8GB
- OS: WINDOWS 10 64bit
- STORAGE : 8GB
- GPU : 2GB

### **2.3.2 SOFTWARE AND TOOLS SPECIFICATION**

- Python3
- Virtualbox
- Scapy
- Pip
- Arista switch/router vEOS
- Arista Aboot iso
- Ubuntu VM

### **3. SYSTEM DESIGN**

#### **3.1 MODULE DESIGN**

**nps.py**

**This is the module to capture and analysing the packets over the network.**

**Scapy is the library used in it , we will use the sniff() function to capture network packets.**

**#To see a list of what functions Scapy has available, open Scapy and run the ls() function.**

**#Run the ls() function to see ALL the supported protocols.**

**#Run the ls(protocol) function to see the fields and default values for any protocol. E.g. ls(BOOTP)**

**#See packet layers and contents with the .show() method.**

**#Dig into a specific packet layer using a list index: pkts[3][2].summary()**

**#...the first index chooses the packet out of the pkts list, the second index chooses the layer for that specific packet.**

**#Using the .command() method will return a string for the command necessary to recreate that sniffed packet.**

**#To see the list of optional arguments for the sniff() function:**

**#print(sniff.\_\_doc\_\_)**

**'''**

**Sniff packets and return a list of packets.**

**Arguments:**

**count: number of packets to capture. 0 means infinity.**

**store: whether to store sniffed packets or discard them**

**prn:** function to apply to each packet. If something is returned, it is displayed.

**Ex:** `prn = lambda x: x.summary()`

**filter:** BPF filter to apply.

**lfilter:** Python function applied to each packet to determine if further action may be done.

**Ex:** `lfilter = lambda x: x.haslayer(Padding)`

**offline:** PCAP file (or list of PCAP files) to read packets from, instead of sniffing them

**timeout:** stop sniffing after a given time (default: None).

**L2socket:** use the provided L2socket (default: use `conf.L2listen`).

**opened\_socket:** provide an object (or a list of objects) ready to use `.recv()` on.

**stop\_filter:** Python function applied to each packet to determine if we have to stop the capture after this packet.

**Ex:** `stop_filter = lambda x: x.haslayer(TCP)`

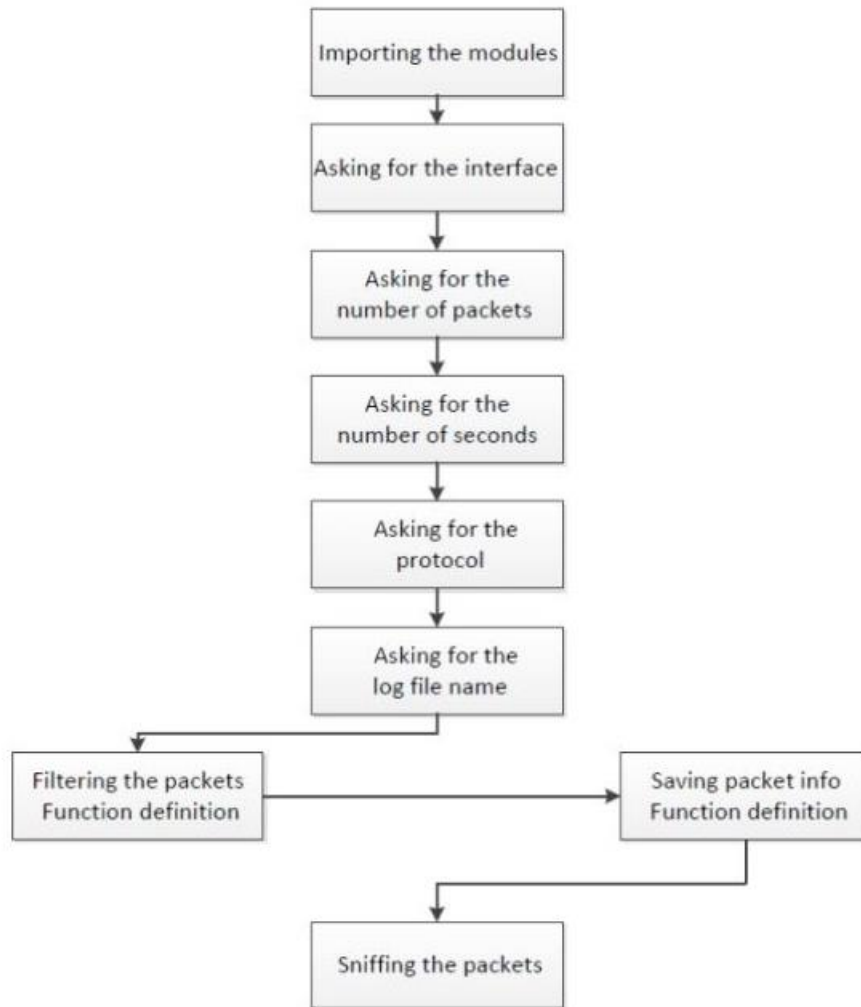
**iface:** interface or list of interfaces (default: None for sniffing on all interfaces).

The `iface`, `offline` and `opened_socket` parameters can be either an





### 3.2 DATA FLOW DIAGRAM



## **4. IMPLEMENTATION, CONFIGURATION AND TESTING**

### **Setting up the working environment**

#### **Installing the Virtualization Software on Windows OS**

Downloading VirtualBox for Windows OS:

<https://www.virtualbox.org/wiki/Downloads>

#### **Downloading necessary files for Arista switch:**

simply go to the EOS Download section at <https://www.arista.com/en/support/software-download> and search for the following:-

- **Abboot-veos-8.0.0.iso**
- **vEOS-lab-4.20.15M.vmdk**

# Configuring the Arista Switch/Router VM

## ✓ Arista 1 VM configuration

The screenshot displays the Oracle VM VirtualBox Manager interface. On the left, a list of VMs includes Arista 1 (Running), Arista 2 (Running), Arista 3 (Powered Off), and UbuntuVM (Running). The main pane shows the configuration for Arista 1, with tabs for General, System, Display, Storage, Audio, Network, USB, Shared folders, and Description. The General tab is active, showing the VM's name, operating system, and settings file location. A preview window on the right shows the VM's display output, which is currently blank.

**Oracle VM VirtualBox Manager**

File Machine Help

**Tools**

New Settings Discard Show

**Arista 1** Running

**Arista 2** Running

**Arista 3** Powered Off

**UbuntuVM** Running

**General**

Name: Arista 1  
Operating System: Fedora (64-bit)  
Settings File Location: C:\Users\mkign\VirtualBox VMs\Arista 1

**System**

Base Memory: 1024 MB  
Boot Order: Floppy, Optical, Hard Disk  
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

**Display**

Video Memory: 16 MB  
Graphics Controller: VMSVGA  
Remote Desktop Server: Disabled  
Recording: Disabled

**Storage**

Controller: IDE  
IDE Secondary Master: [Optical Drive] Abboot-veos-8.0.0.iso (5.00 MB)  
Controller: SATA  
SATA Port 0: vEOS-lab-4.20.15M.vmdk (Normal, 4.00 GB)

**Audio**

Disabled

**Network**

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Microsoft KM-TEST Loopback Adapter)

**USB**

USB Controller: OHCI  
Device Filters: 0 (0 active)

**Shared folders**

None

**Description**

None

**Preview**

```
Arista login> admin
Arista#
```

## ✓ Arista 2 VM configuration

The screenshot displays the Oracle VM VirtualBox Manager interface. On the left, a list of VMs includes Arista 1 (Running), Arista 2 (Running), Arista 3 (Powered Off), and UbuntuVM (Running). The main pane shows the configuration for Arista 2, with tabs for General, System, Display, Storage, Audio, Network, USB, Shared folders, and Description. The General tab is active, showing the VM's name, operating system, and settings file location. A Preview window shows a terminal with login prompts.

Oracle VM VirtualBox Manager

File Machine Help

Tools

New Settings Discard Show

**Arista 1** Running

**Arista 2** Running

**Arista 3** Powered Off

**UbuntuVM** Running

**General**

Name: Arista 2  
Operating System: Fedora (64-bit)  
Settings File Location: C:\Users\mkign\VirtualBox VMs\Arista 2

**System**

Base Memory: 1024 MB  
Boot Order: Floppy, Optical, Hard Disk  
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

**Display**

Video Memory: 16 MB  
Graphics Controller: VMSVGA  
Remote Desktop Server: Disabled  
Recording: Disabled

**Storage**

Controller: IDE  
IDE Secondary Master: [Optical Drive] Aboot-veos-8.0.0.iso (5.00 MB)  
Controller: SATA  
SATA Port 0: Arista 2-disk1.vmdk (Normal, 4.00 GB)

**Audio**

Disabled

**Network**

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Microsoft KM-TEST Loopback Adapter)

**USB**

USB Controller: OHCI  
Device Filters: 0 (0 active)

**Shared folders**

None

**Description**

None

**Preview**

```
Arista2 login: admin
Password:
Arista2#enable
Arista2(config)#terminal
Arista2(config)#
```

## ✓ Arista 3 VM configuration

The screenshot displays the Oracle VM VirtualBox Manager interface. On the left, a list of VMs includes Arista 1 (Running), Arista 2 (Running), Arista 3 (Powered Off), and UbuntuVM (Running). The main pane shows the configuration for Arista 3, which is currently powered off. The configuration is organized into several sections: General, System, Display, Storage, Audio, Network, USB, Shared folders, and Description. The General section shows the VM name as 'Arista 3', the operating system as 'Fedora (64-bit)', and the settings file location. The System section shows a base memory of 1024 MB and boot order settings. The Display section shows 16 MB of video memory and disabled remote desktop and recording. The Storage section shows an IDE controller with a secondary master drive containing the bootable ISO file. The Network section shows a single Intel PRO/1000 MT Desktop network adapter. The Audio, USB, Shared folders, and Description sections are currently set to 'None'.

Oracle VM VirtualBox Manager

File Machine Help

Tools

New Settings Discard Start

**Arista 1** Running

**Arista 2** Running

**Arista 3** Powered Off

**UbuntuVM** Running

**General**

Name: Arista 3  
Operating System: Fedora (64-bit)  
Settings File Location: C:\Users\mkign\VirtualBox VMs\Arista 3

**System**

Base Memory: 1024 MB  
Boot Order: Floppy, Optical, Hard Disk  
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

**Display**

Video Memory: 16 MB  
Graphics Controller: VMSVGA  
Remote Desktop Server: Disabled  
Recording: Disabled

**Storage**

Controller: IDE  
IDE Secondary Master: [Optical Drive] Aboot-veos-8.0.0.iso (5.00 MB)  
Controller: SATA  
SATA Port 0: Arista 3-disk1.vmdk (Normal, 4.00 GB)

**Audio**

Disabled

**Network**

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Microsoft KM-TEST Loopback Adapter)

**USB**

USB Controller: OHCI  
Device Filters: 0 (0 active)

**Shared folders**

None

**Description**

None

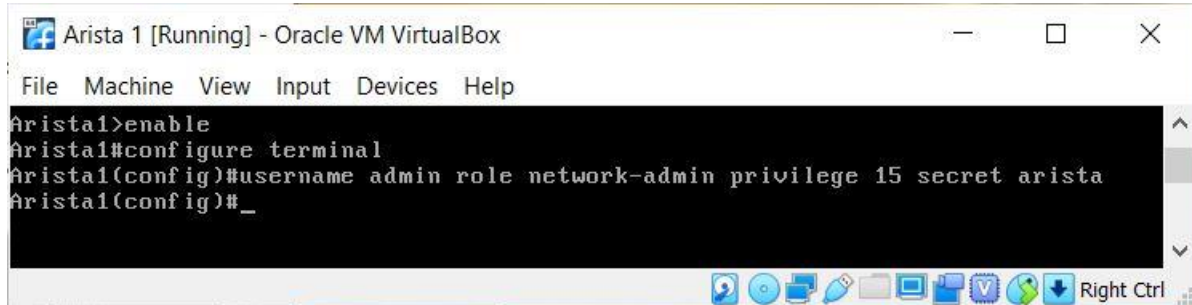
**Preview**

Arista 3

## Necessary Switch/Router Configuration

### 1. Arista1

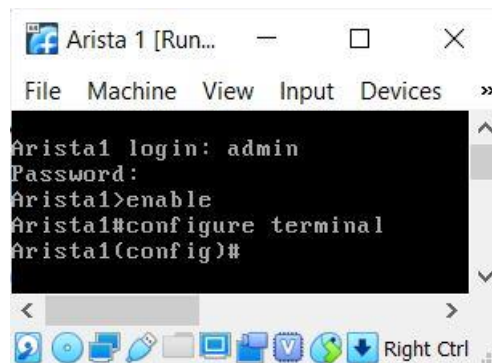
- ✓ **Configuring the user and password for Arista1**



A screenshot of a terminal window titled "Arista 1 [Running] - Oracle VM VirtualBox". The terminal shows the following commands and output:

```
Arista1>enable
Arista1#configure terminal
Arista1(config)#username admin role network-admin privilege 15 secret arista
Arista1(config)#_
```

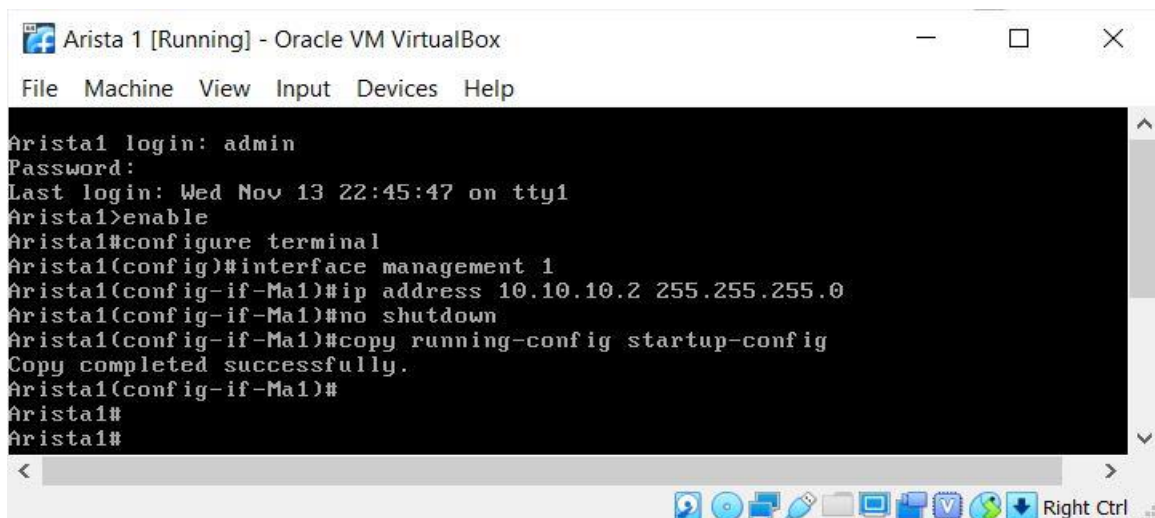
- ✓ **Now exit the switch/router by using exit command and login the device again**



A screenshot of a terminal window titled "Arista 1 [Run...". The terminal shows the following commands and output:

```
Arista1 login: admin
Password:
Arista1>enable
Arista1#configure terminal
Arista1(config)#
```

- ✓ **Configuring Arista1 interface ip address and save the configuration by using command- 'copy running-config startup-config'**

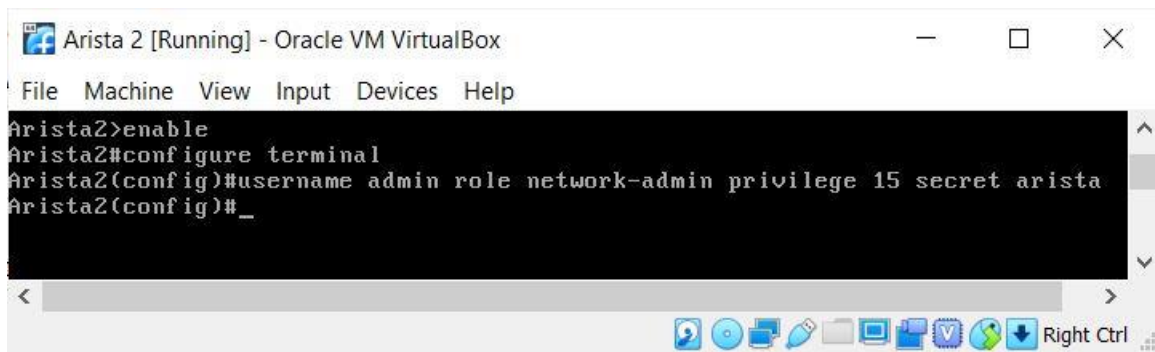


A screenshot of a terminal window titled "Arista 1 [Running] - Oracle VM VirtualBox". The terminal shows the following commands and output:

```
Arista1 login: admin
Password:
Last login: Wed Nov 13 22:45:47 on tty1
Arista1>enable
Arista1#configure terminal
Arista1(config)#interface management 1
Arista1(config-if-Ma1)#ip address 10.10.10.2 255.255.255.0
Arista1(config-if-Ma1)#no shutdown
Arista1(config-if-Ma1)#copy running-config startup-config
Copy completed successfully.
Arista1(config-if-Ma1)#
Arista1#
Arista1#
```

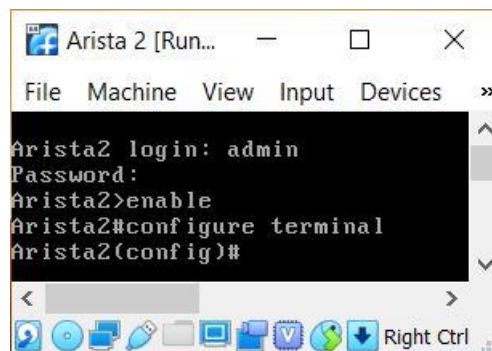
## 2. Arista2

### ✓ Configuring the user and password for Arista2



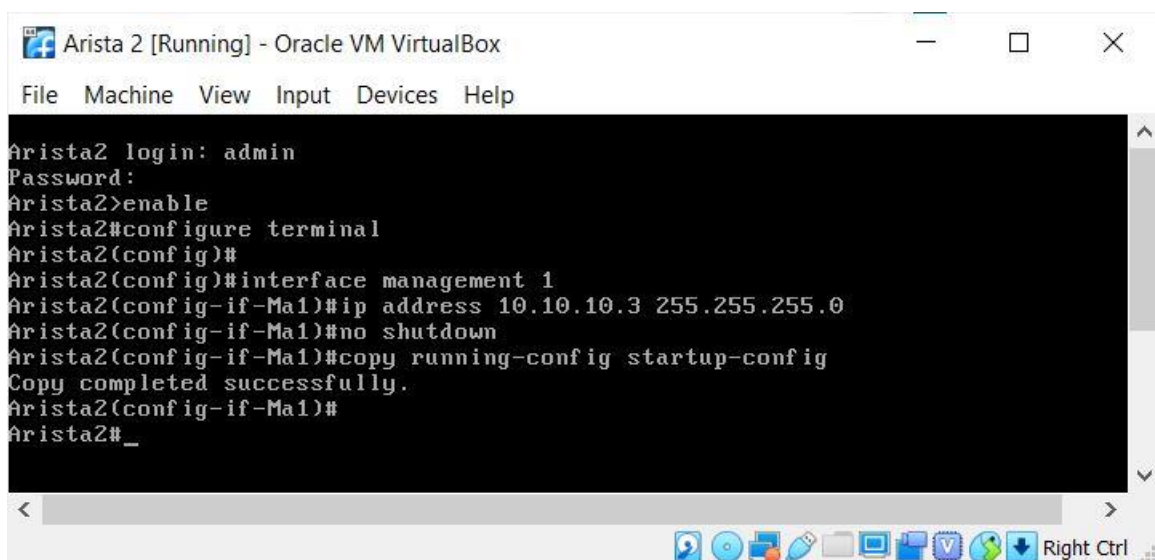
```
Arista2>enable
Arista2#configure terminal
Arista2(config)#username admin role network-admin privilege 15 secret arista
Arista2(config)#_
```

### ✓ Now exit the switch/router by using exit command and login the device again



```
Arista2 login: admin
Password:
Arista2>enable
Arista2#configure terminal
Arista2(config)#
```

### ✓ Configuring Arista2 interface ip address and save the configuration by using command- 'copy running-config startup-config'

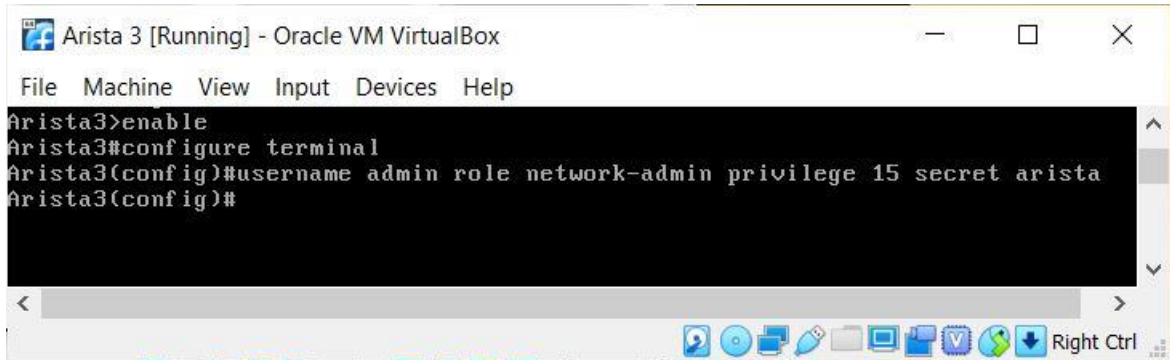


```
Arista2 login: admin
Password:
Arista2>enable
Arista2#configure terminal
Arista2(config)#
Arista2(config)#interface management 1
Arista2(config-if-Ma1)#ip address 10.10.10.3 255.255.255.0
Arista2(config-if-Ma1)#no shutdown
Arista2(config-if-Ma1)#copy running-config startup-config
Copy completed successfully.
Arista2(config-if-Ma1)#
Arista2#_
```



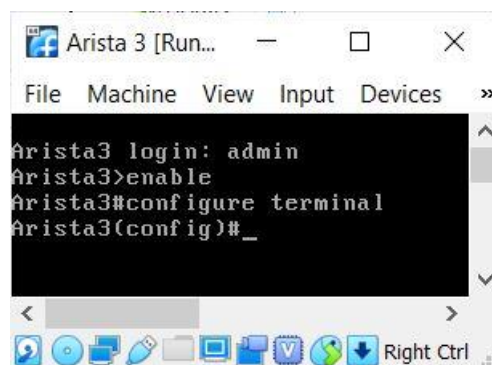
### 3. Arista3

#### ✓ Configuring the user and password for Arista3



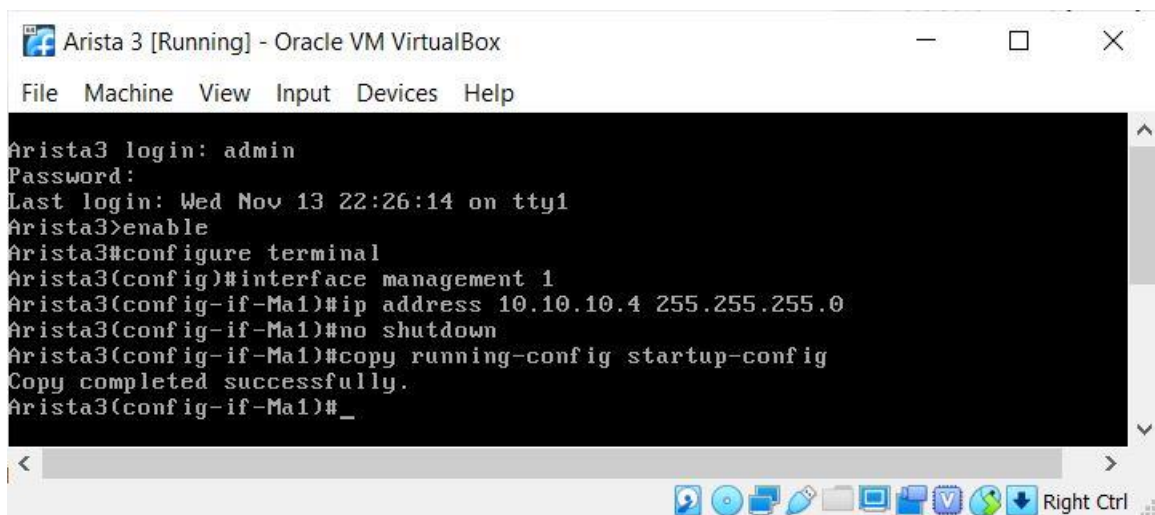
```
Arista3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Arista3>enable
Arista3#configure terminal
Arista3(config)#username admin role network-admin privilege 15 secret arista
Arista3(config)#
```

#### ✓ Now exit the switch/router by using exit command and login the device again



```
Arista3 [Run...
File Machine View Input Devices >>
Arista3 login: admin
Arista3>enable
Arista3#configure terminal
Arista3(config)#_
```

#### ✓ Configuring Arista3 interface ip address and save the configuration by using command- 'copy running-config startup-config'



```
Arista3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Arista3 login: admin
Password:
Last login: Wed Nov 13 22:26:14 on tty1
Arista3>enable
Arista3#configure terminal
Arista3(config)#interface management 1
Arista3(config-if-Ma1)#ip address 10.10.10.4 255.255.255.0
Arista3(config-if-Ma1)#no shutdown
Arista3(config-if-Ma1)#copy running-config startup-config
Copy completed successfully.
Arista3(config-if-Ma1)#_
```



# Required resource & it's configuration to develop the network packet sniffer application

## Downloading the Linux VM

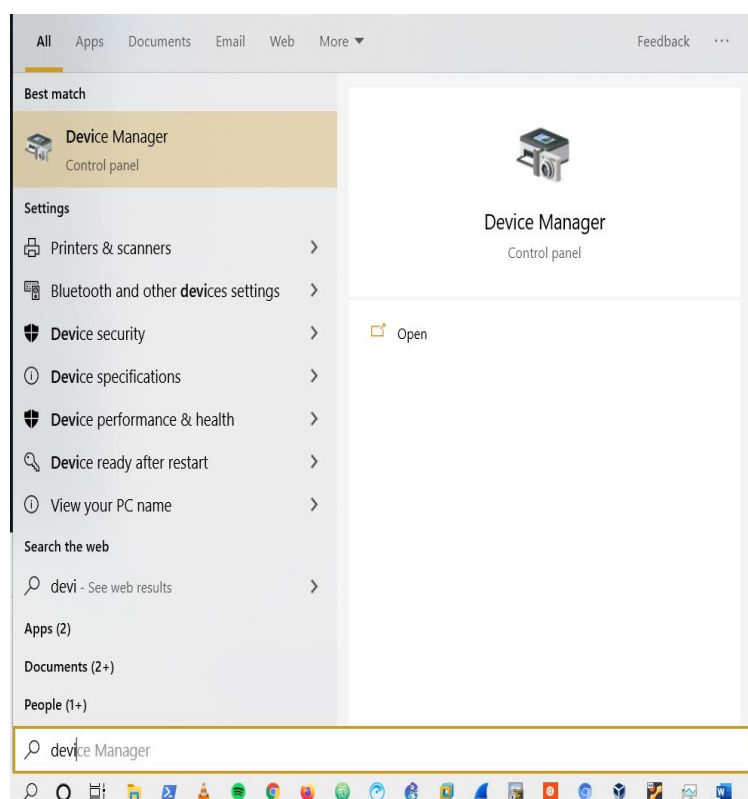
Download the Ubuntu 18.04 LTS .iso file:

Official link: <https://www.ubuntu.com/download/desktop>

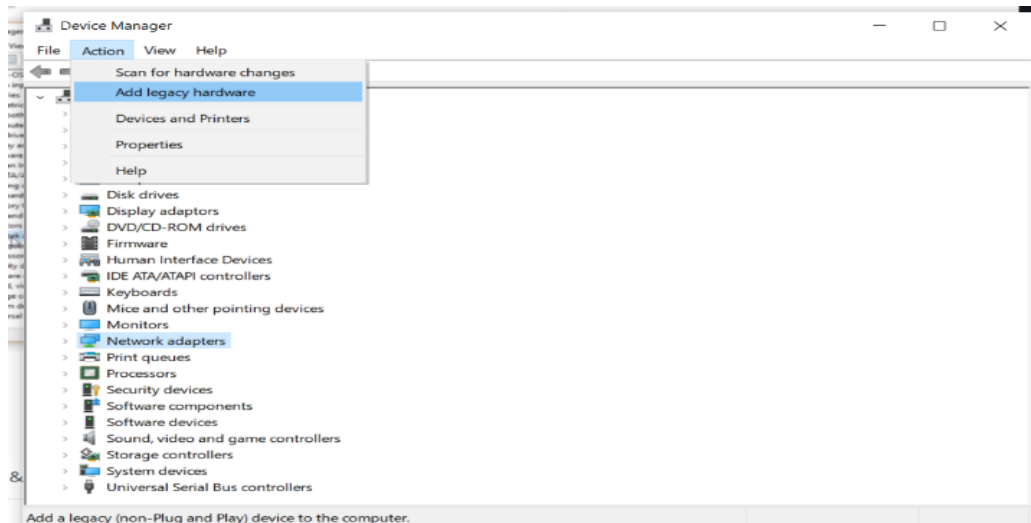
## Connecting the local PC to the device

Steps to Configuring the network adapter: Microsoft KM-TEST loopback adapter on local system

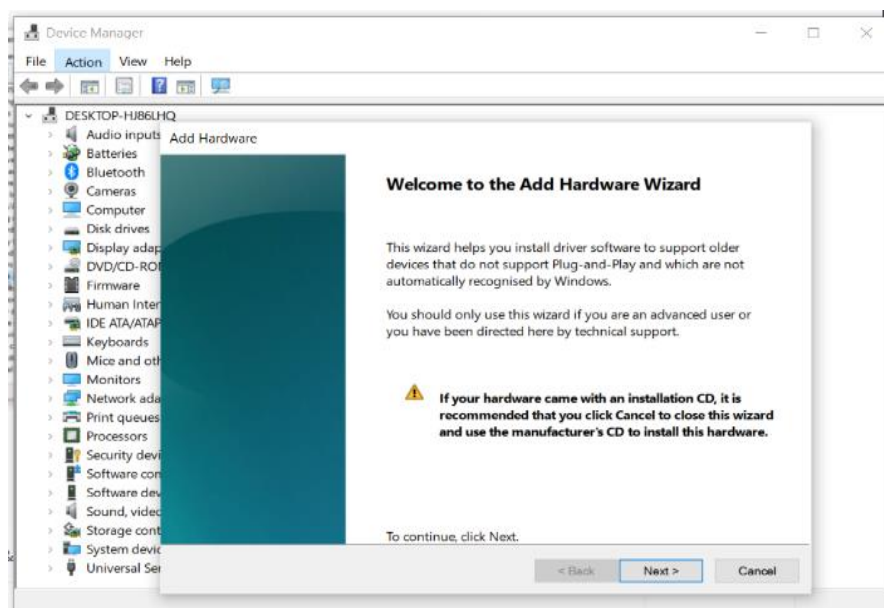
- ✓ Open the device manager from start menu on local system(Windows 10)



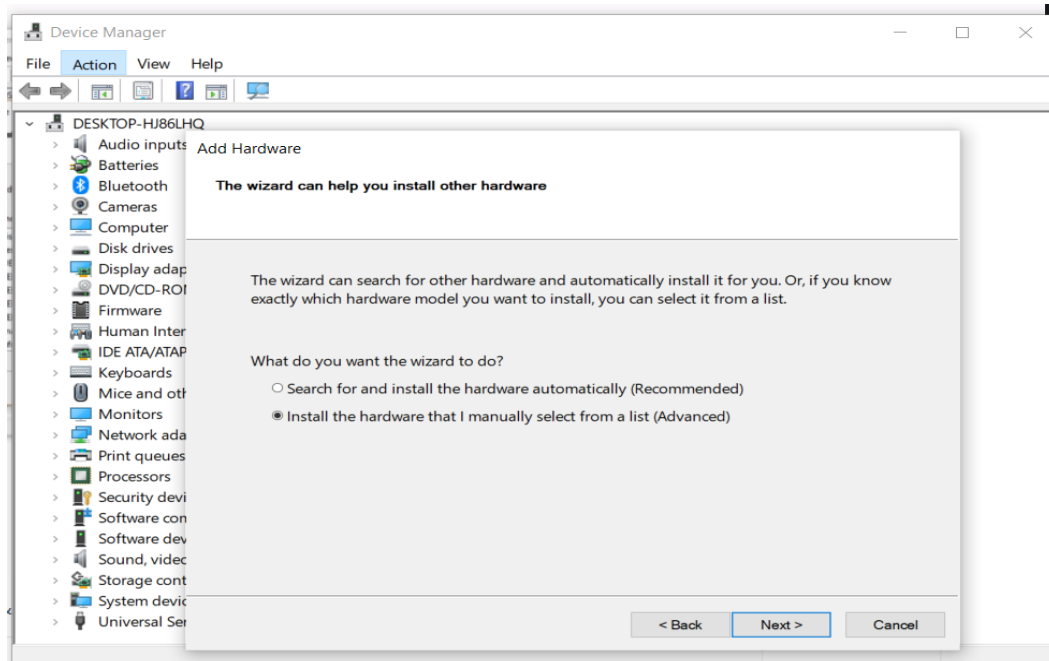
- ✓ Select the network adapter and add the legacy hardware in action menu



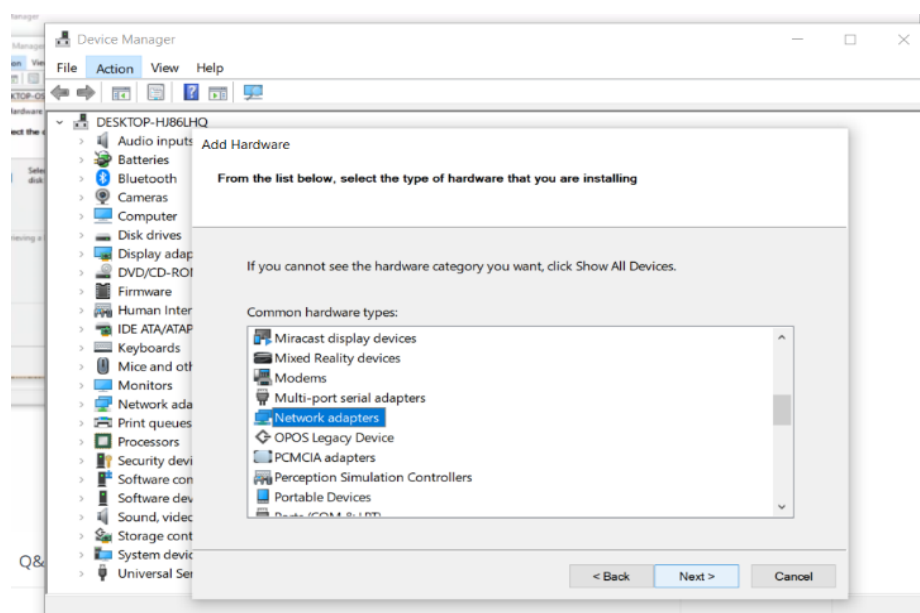
- ✓ Proceed to next step by click next



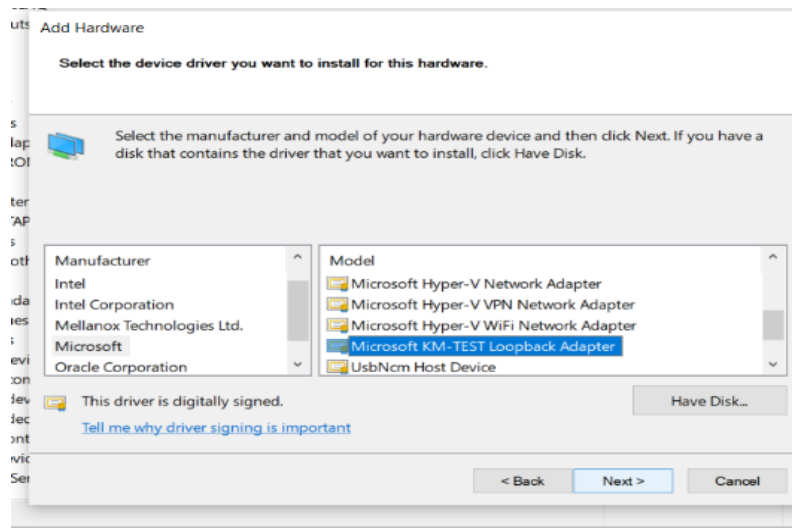
- ✓ Now select second option(Install the hardware that I manually select from a list) and click next



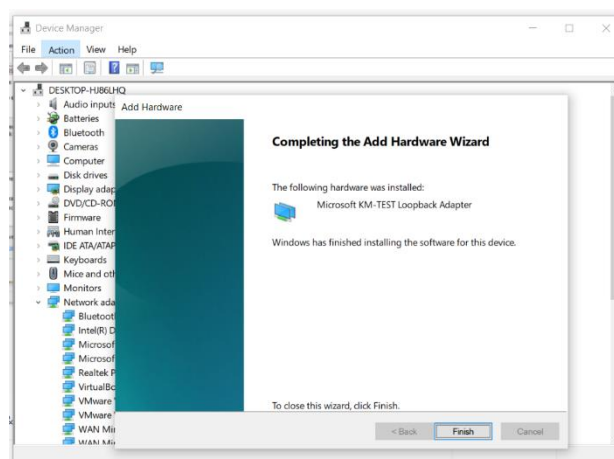
- ✓ Select network adapter and click next



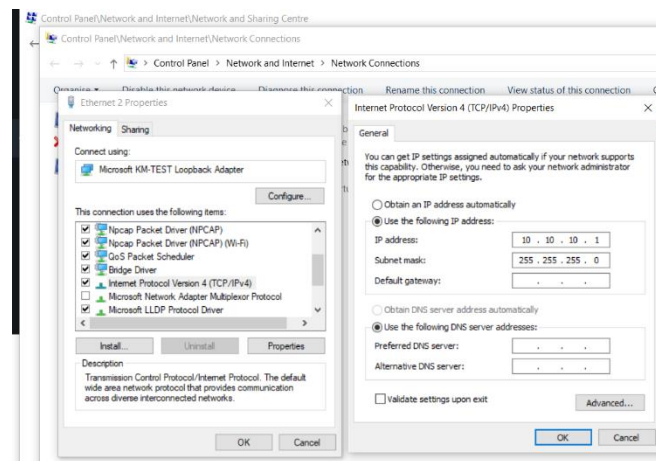
- ✓ **Select Microsoft from left side and Microsoft KM-Test Loopback Adapter and click next**



- ✓ **Then click Finish**



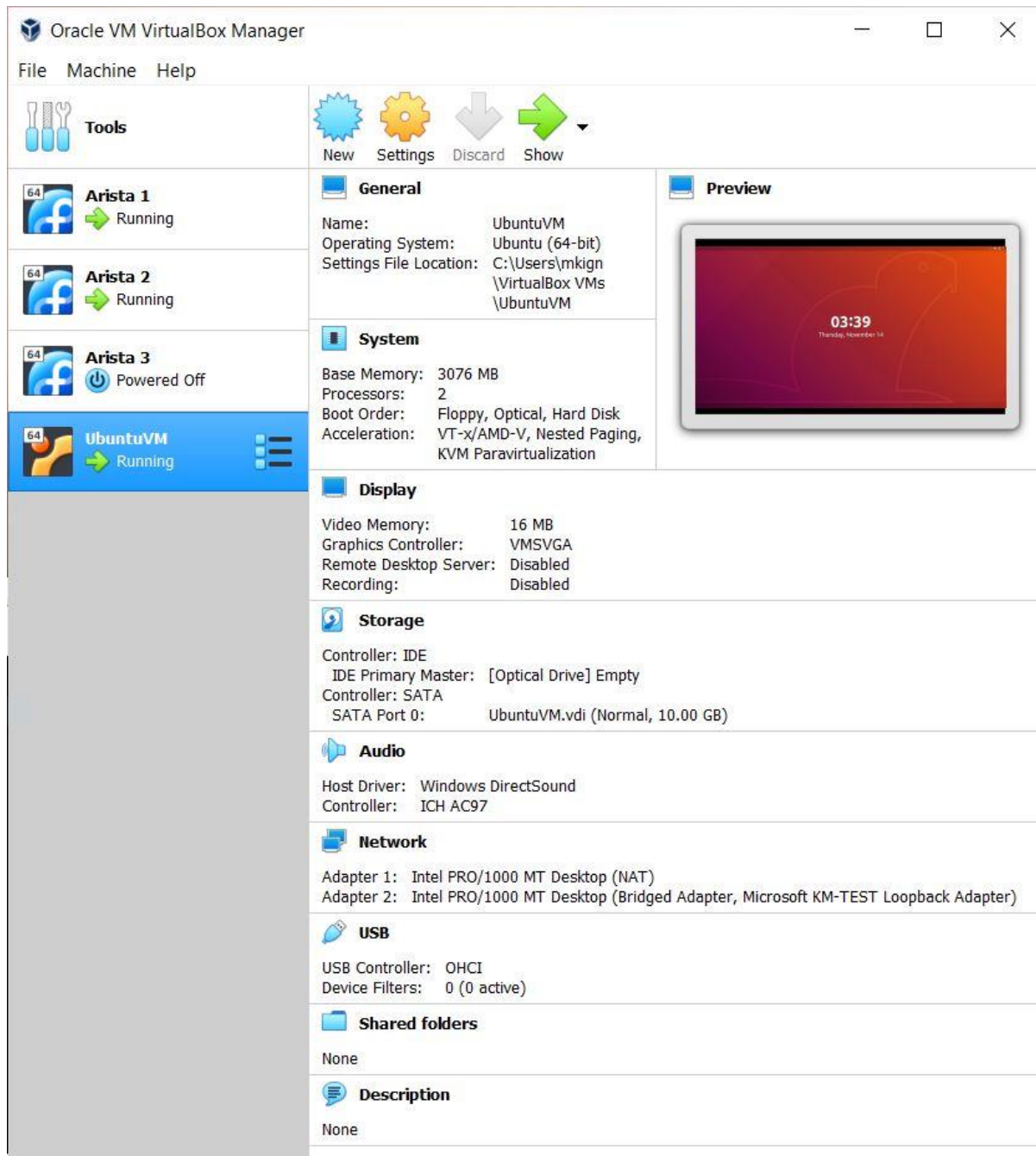
- ✓ **Configuring the ip address for the above adapter**



# Configuring the Linux VM

**Step 1 - VirtualBox settings for the Ubuntu 18.04 VM** (in VirtualBox select the Ubuntu VM and click on the **Settings** button):

- **System** tab - Base memory: 3076 MB (preferably)
- **Network** tab - **Adapter 1:** NAT, **Adapter 2:** Bridged Adapter (then choose your Microsoft loopback adapter)

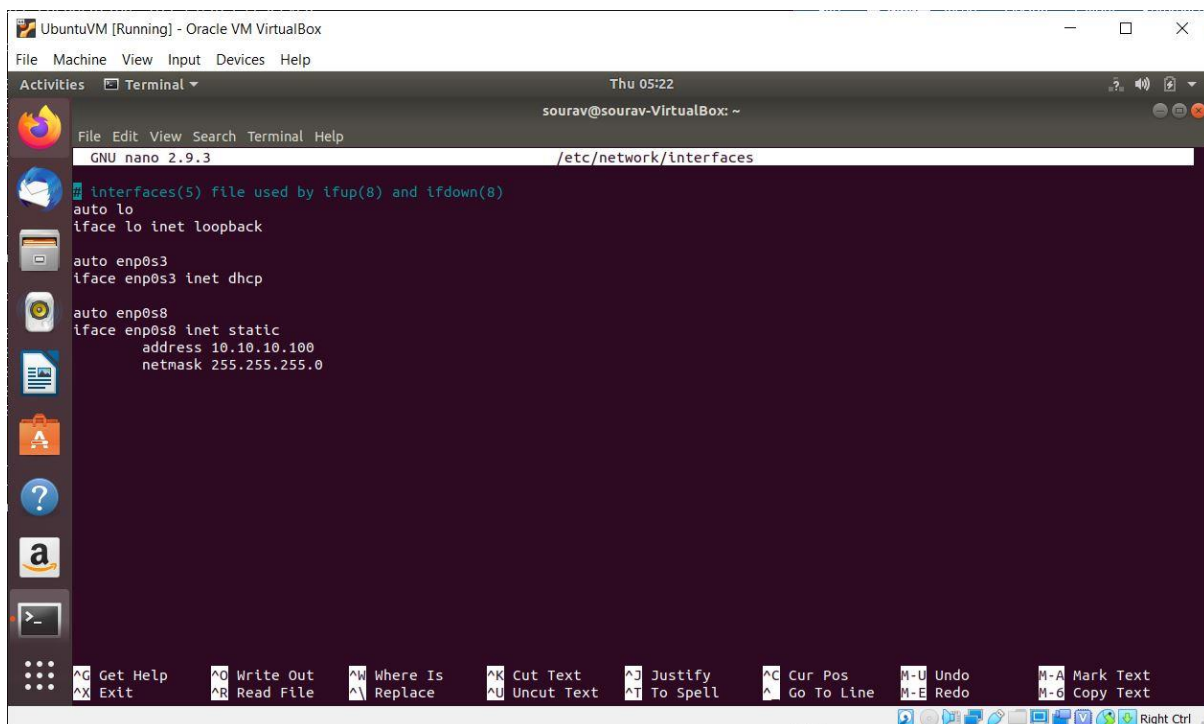


## Step 2 - Ubuntu settings in the Linux Terminal:

- ***cat /etc/network/interfaces*** - showing the current network settings
- ***sudo nano /etc/network/interfaces*** - editing the network settings, making them available after reboot (use the root password that you configured at installation time)
- **use Ctrl+O, hit Enter to confirm, then use Ctrl+X** to save your settings and exit nano.

## Step 3 - /etc/network/interfaces settings to add to the default file:

1. auto enp0s3
2. iface enp0s3 inet dhcp
- 3.
4. auto enp0s8
5. iface enp0s8 inet static
6.     address 10.10.10.100
7.     netmask 255.255.255.0



The screenshot shows a terminal window titled 'UbuntuVM [Running] - Oracle VM VirtualBox'. The terminal is running the 'nano' text editor, editing the file '/etc/network/interfaces'. The current content of the file is as follows:

```
interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet dhcp

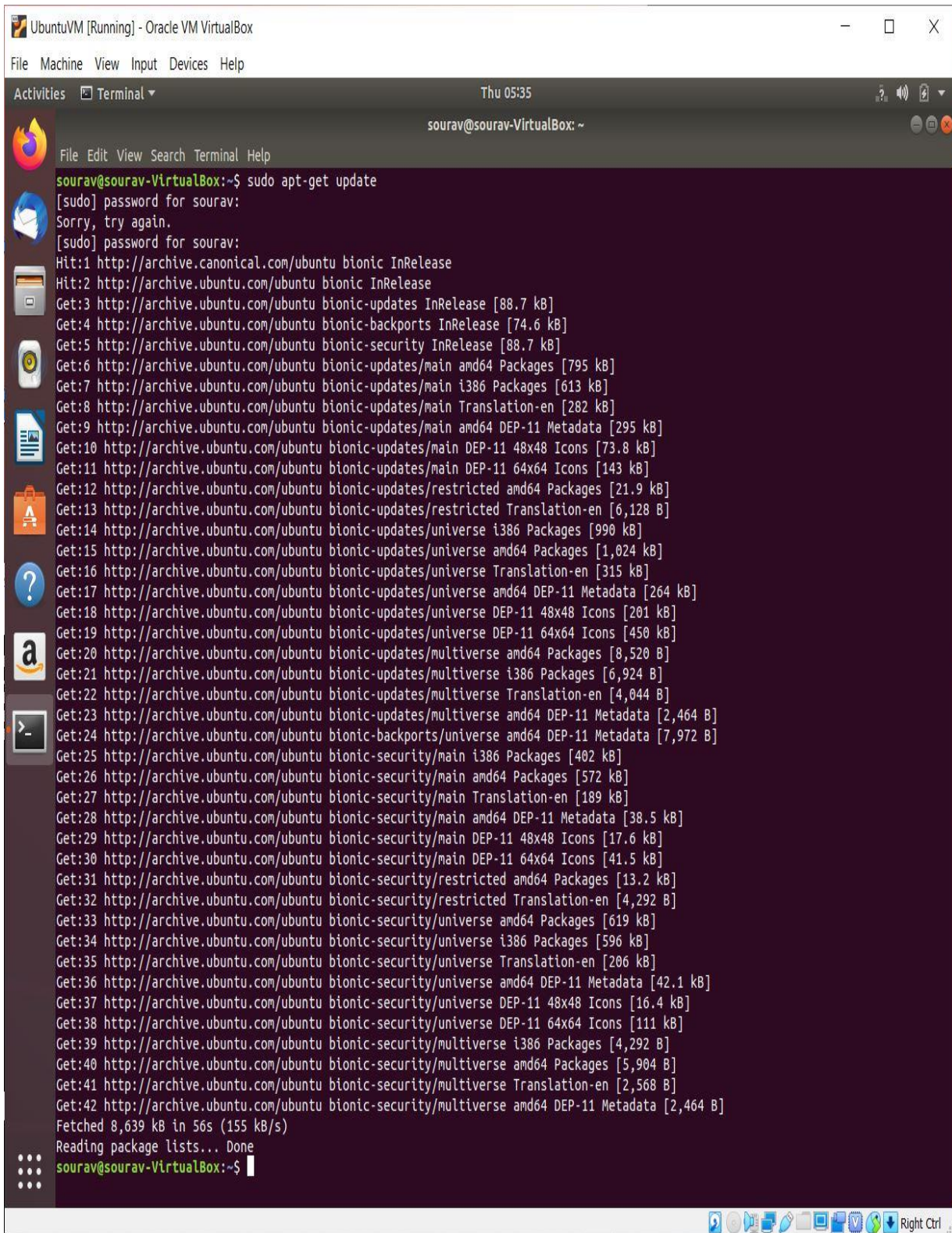
auto enp0s8
iface enp0s8 inet static
    address 10.10.10.100
    netmask 255.255.255.0
```

The terminal window also shows a sidebar with various application icons and a bottom status bar with keyboard shortcuts like 'Get Help', 'Write Out', 'Where Is', etc.



## Step 4 - Update the package lists

- **sudo apt-get update** - run in Terminal, use the root password that we configured at installation time

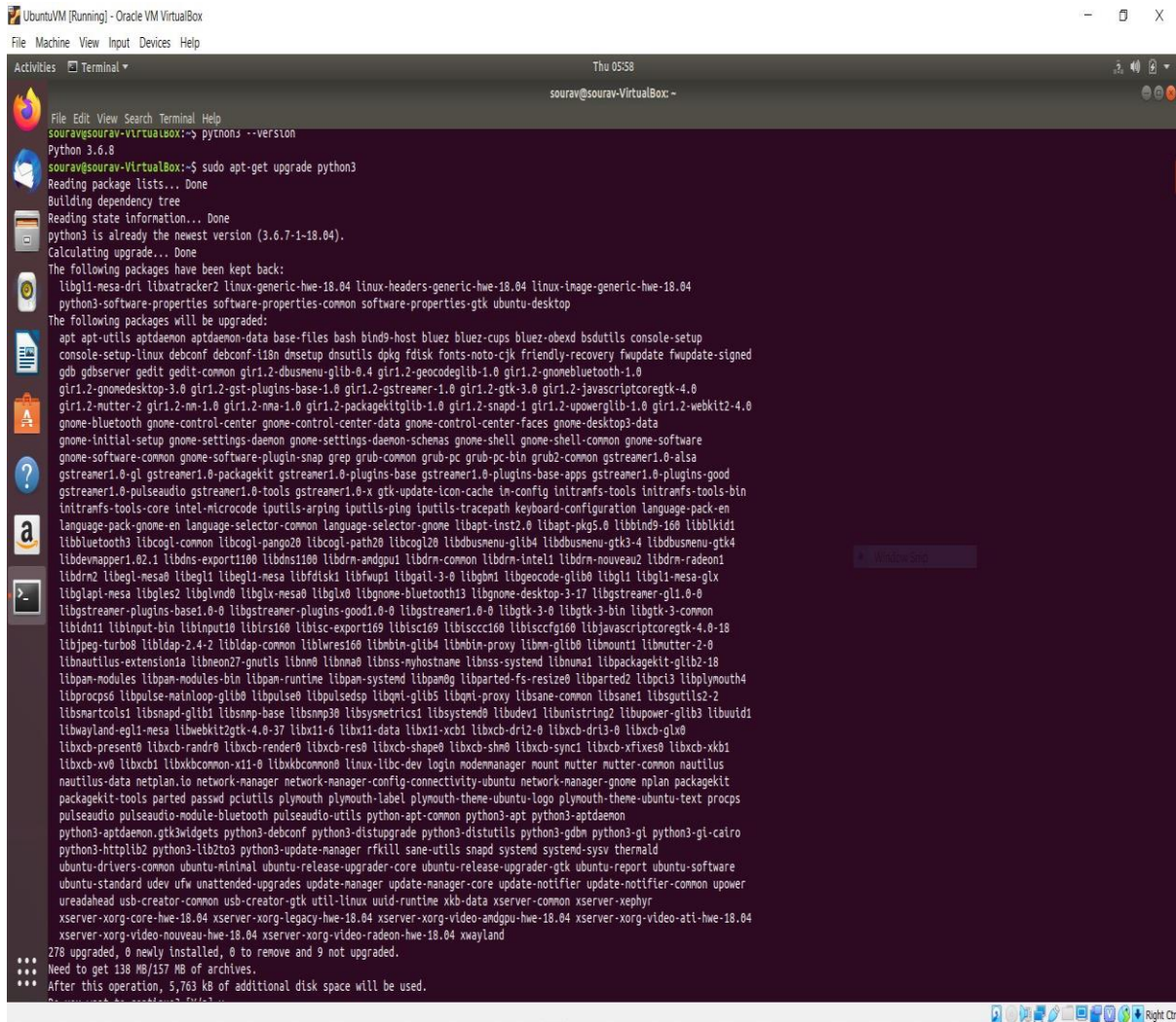


```
UbuntuVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thu 05:35
sourav@sourav-VirtualBox: ~

sourav@sourav-VirtualBox:~$ sudo apt-get update
[sudo] password for sourav:
Sorry, try again.
[sudo] password for sourav:
Hit:1 http://archive.canonical.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [795 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [613 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [282 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [295 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 48x48 Icons [73.8 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 64x64 Icons [143 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [21.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/restricted Translation-en [6,128 B]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [990 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1,024 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [315 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [264 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 48x48 Icons [201 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 64x64 Icons [450 kB]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [8,520 B]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse i386 Packages [6,924 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse Translation-en [4,044 B]
Get:23 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [7,972 B]
Get:25 http://archive.ubuntu.com/ubuntu bionic-security/main i386 Packages [402 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic-security/main amd64 Packages [572 kB]
Get:27 http://archive.ubuntu.com/ubuntu bionic-security/main Translation-en [189 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [38.5 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-security/main DEP-11 48x48 Icons [17.6 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-security/main DEP-11 64x64 Icons [41.5 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [13.2 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-security/restricted Translation-en [4,292 B]
Get:33 http://archive.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [619 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic-security/universe i386 Packages [596 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic-security/universe Translation-en [206 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [42.1 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-security/universe DEP-11 48x48 Icons [16.4 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic-security/universe DEP-11 64x64 Icons [111 kB]
Get:39 http://archive.ubuntu.com/ubuntu bionic-security/multiverse i386 Packages [4,292 B]
Get:40 http://archive.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [5,904 B]
Get:41 http://archive.ubuntu.com/ubuntu bionic-security/multiverse Translation-en [2,568 B]
Get:42 http://archive.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Fetched 8,639 kB in 56s (155 kB/s)
Reading package lists... Done
sourav@sourav-VirtualBox:~$
```

## Step 5 - Upgrade Python 3

- **python3 --version** - run in Terminal to see the current Python 3 version
- **sudo apt-get upgrade python3** - upgrade Python to the latest version (may take a couple of minutes to finish)



```
UbuntuVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Thu 05:58
sourav@sourav-VirtualBox: ~
sourav@sourav-VirtualBox:~$ python3 --version
Python 3.6.8
sourav@sourav-VirtualBox:~$ sudo apt-get upgrade python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.6.7-1-18.04).
Calculating upgrade... Done

The following packages have been kept back:
  libgl1-mesa-dri libxatracker2 linux-generic-hwe-18.04 linux-headers-generic-hwe-18.04 linux-image-generic-hwe-18.04
  python3-software-properties software-properties-common software-properties-gtk ubuntu-desktop

The following packages will be upgraded:
  apt apt-utils aptdaemon-data base-files bash bind9-host bluez bluez-cups bluez-obex bsdtar console-setup
  console-setup-linux debconf debconf-i18n dnsetup dnsutils dpkg fdisk fonts-noto-cjk friendly-recovery fwupdate fwupdate-signed
  gdb gobservers gedit gedit-common gir1.2-dbusmenu-glib-0.4 gir1.2-geocodeglib-1.0 gir1.2-gnomebluetooth-1.0
  gir1.2-gnomedesktop-3.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gtk-3.0 gir1.2-javascriptcoregtk-4.0
  gir1.2-mutter-2 gir1.2-nm-1.0 gir1.2-nma-1.0 gir1.2-packagekitglib-1.0 gir1.2-snapd-1 gir1.2-upowerglib-1.0 gir1.2-webkit2-4.0
  gnome-bluetooth gnome-control-center gnome-control-center-data gnome-control-center-faces gnome-desktop3-data
  gnome-initial-setup gnome-settings-daemon gnome-settings-daemon-schemas gnome-shell gnome-shell-common gnome-software
  gnome-software-common gnome-software-plugin-snap grep grub-common grub-pc grub-pc-bin grub2-common gstreamer1.0-alsa
  gstreamer1.0-gi gstreamer1.0-packagekit gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-good
  gstreamer1.0-pulseaudio gstreamer1.0-tools gstreamer1.0-x gtk-update-icon-cache ln-config initramfs-tools initramfs-tools-bin
  initramfs-tools-core intel-microcode iputils-arping iputils-ping iputils-tracepath keyboard-configuration language-pack-en
  language-pack-gnome-en language-selector-common language-selector-gnome libapt-inst2.0 libapt-pkg5.0 libbind9-160 libblkid1
  libbluetooth3 libbogl-common libbogl-pango2 libbogl-path20 libbogl20 libdbusmenu-glib4 libdbusmenu-gtk3-4 libdbusmenu-gtk4
  libdevmapper1.02.1 libdns-export1100 libdns1100 libdrm-andgpus1 libdrm-common libdrm-intel1 libdrm-nouveau2 libdrm-radeon1
  libdrm2 libegl-mesa0 libegl1 libegl1-mesa libfdisk1 libfwupd1 libgail-3.0 libgbm1 libgeocode-glib0 libgl1 libgl1-mesa-glx
  libglapi-mesa libgles2 libglvnd0 libglx-mesa0 libglx0 libgnome-bluetooth13 libgnome-desktop-3-17 libgstreamer-glib1.0-0
  libgstreamer-plugins-base1.0-0 libgstreamer-plugins-good1.0-0 libgstreamer1.0-0 libgtk-3-0 libgtk-3-bin libgtk-3-common
  libidn11 libinput-bin libinput10 liblrs100 liblsc-export109 liblsc109 liblsc100 liblsc100 liblsc100 liblsc100 liblsc100
  libjpeg-turbo8 libldap-2.4-2 libldap-common liblws100 libmbin-glib4 libmbin-proxy libmbn-glib0 libmount1 libmutter-2-0
  libnautilus-extension1a libneon27-gnutls libnm0 libnm0 libnss-myhostname libnss-systemd libnuna1 libpackagekit-glib2-18
  libpan-modules libpan-runtime libpan-system libpan0 libparted-fs-resize0 libparted2 libpc13 libplymouth4
  libprocps0 libpulse-mainloop-glib0 libpulse0 libpulse0 libqmi-glib5 libqmi-proxy libsane-common libsane1 libsgutils2-2
  libsmartcols1 libsnapd-glib1 libsnmp-base libsnmp30 libsysmetrics1 libsystemd0 libudev1 libunistring2 libupower-glib3 libuuid1
  libwayland-egl1-mesa libwebkit2gtk-4.0-37 libx11-6 libx11-data libx11-xcb libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
  libxcb-present0 libxcb-randr0 libxcb-render0 libxcb-res0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb-xxkb1
  libxcb-xxkb libxcb-xxkb-common-x11-0 libxcb-common0 linux-libc-dev login nodermanager mount mutter mutter-common nautilus
  nautilus-data netplan.io network-manager network-manager-config-connectivity-ubuntu network-manager-gnome nplan packagekit
  packagekit-tools parted passwd pcutils plymouth plymouth-label plymouth-theme-ubuntu-logo plymouth-theme-ubuntu-text procs
  pulseaudio pulseaudio-module-bluetooth pulseaudio-utils python-apt-common python3-apt python3-aptdaemon
  python3-aptdaemon.gtk3widgets python3-debconf python3-distupgrade python3-distutils python3-gdbm python3-gi python3-gi-cairo
  python3-httplib2 python3-libz3 python3-update-manager rkill sane-utils snapd systemd systemd-sysv thermald
  ubuntu-drivers-common ubuntu-minimal ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk ubuntu-report ubuntu-software
  ubuntu-standard udev ufw unattended-upgrades update-manager update-manager-core update-notifier update-notifier-common upower
  ureadahead usb-creator-common usb-creator-gtk util-linux uuid-runtime xkb-data xserver-common xserver-xephyr
  xserver-xorg-core-hwe-18.04 xserver-xorg-legacy-hwe-18.04 xserver-xorg-video-andgpus-hwe-18.04 xserver-xorg-video-ati-hwe-18.04
  xserver-xorg-video-nouveau-hwe-18.04 xserver-xorg-video-radeon-hwe-18.04 xwayland

278 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
Need to get 138 MB/157 MB of archives.
After this operation, 5,763 kB of additional disk space will be used.
```

## Step 6 - Install net-tools (and ifconfig) and the pip package manager

- **sudo apt install net-tools** - run in Terminal, use the root password that we configured at installation time
- use the **ifconfig** command in the Terminal to see all the network interfaces
- **sudo apt install python3-pip** - run in Terminal, use the root password that we configured at installation time

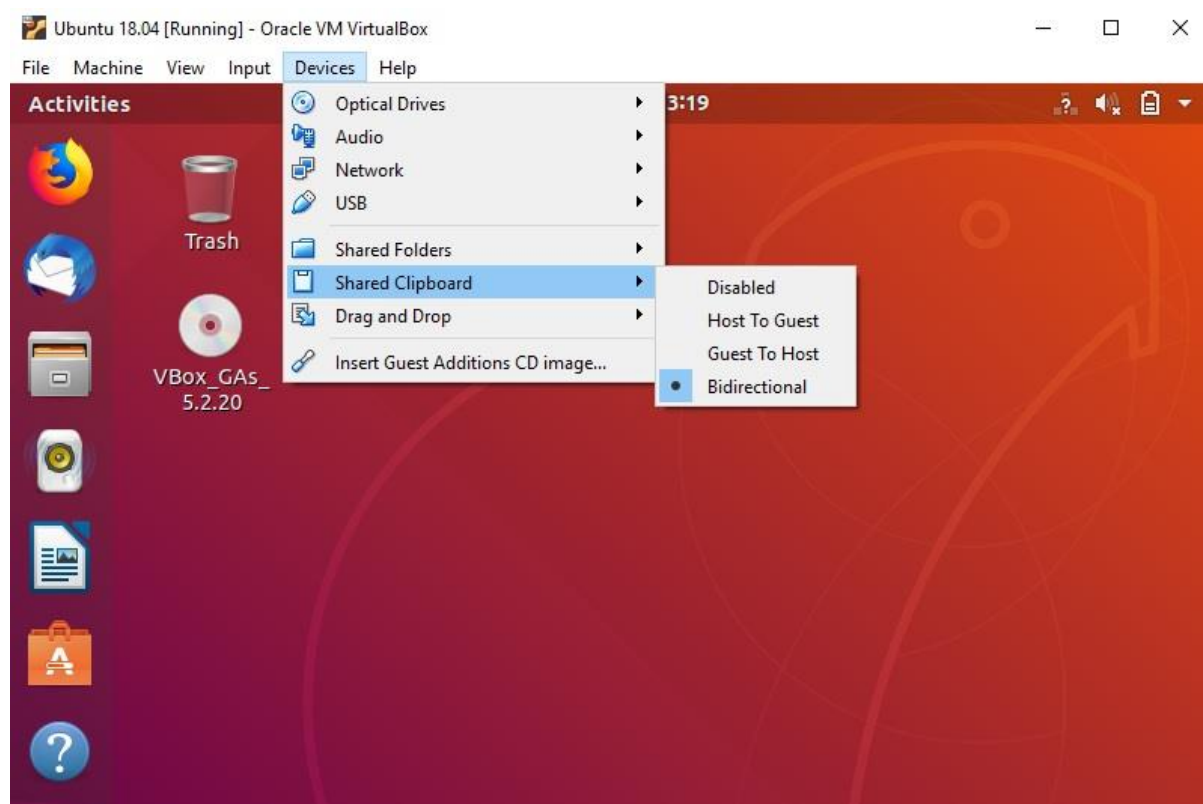


## Step 7 - Install Scapy

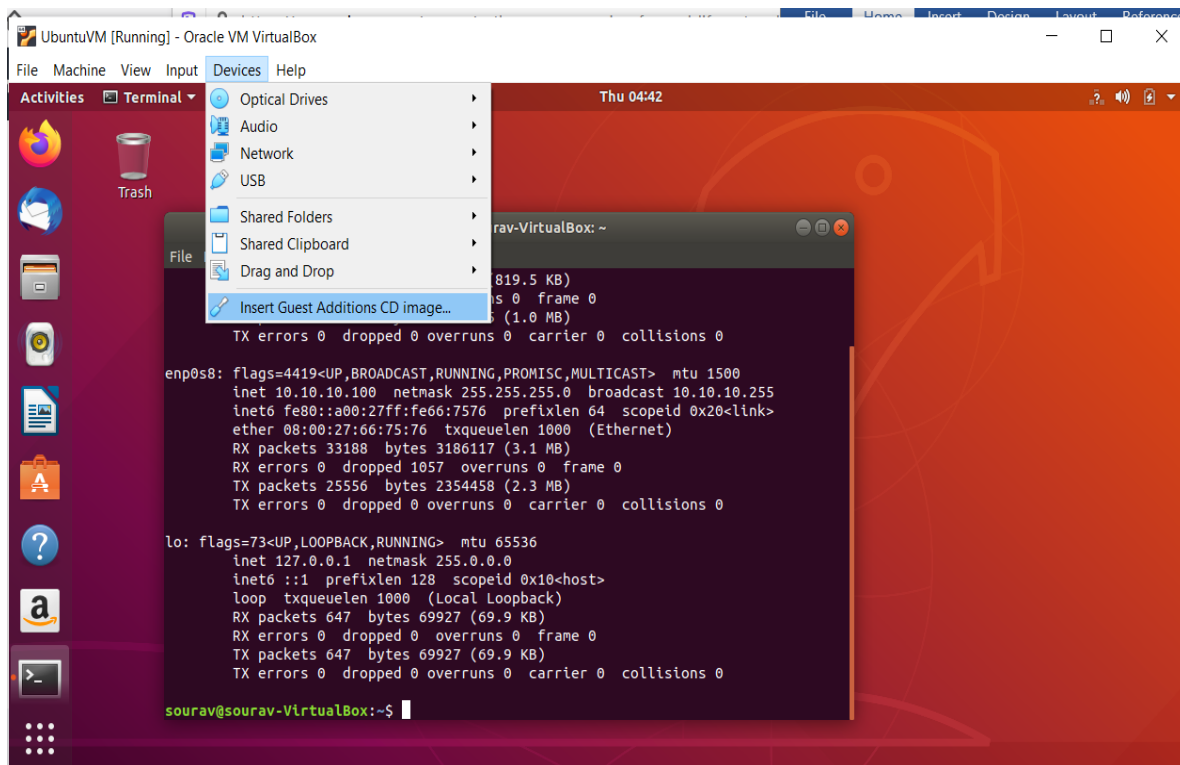
- **sudo pip3 install scapy** - run in Terminal, use the root password that we configured at installation time
- Scapy's official documentation available at <https://scapy.readthedocs.io/en/latest/>

**Step 8 (optional) - Install VirtualBox Guest Additions** to enable shared clipboard (copying and pasting between the host OS and guest OS) and other useful features.

- Having your Ubuntu VM open, click on **Devices - Shared Clipboard** and select **Bidirectional**.



- Having our Ubuntu VM open, click on **Devices - Install Guest Additions CD Image**, then click on **Run**, enter your root password and wait for the process to finish. Finally, press **Enter** when prompted and **reboot** the Ubuntu VM.



## Testing the connections from local devices to network devices(Switch/router)

```
Command Prompt
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\mkign>ping 10.10.10.2

Pinging 10.10.10.2 with 32 bytes of data:
Reply from 10.10.10.2: bytes=32 time<1ms TTL=64
Reply from 10.10.10.2: bytes=32 time<1ms TTL=64
Reply from 10.10.10.2: bytes=32 time<1ms TTL=64
Reply from 10.10.10.2: bytes=32 time<1ms TTL=64

Ping statistics for 10.10.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\mkign>ping 10.10.10.3

Pinging 10.10.10.3 with 32 bytes of data:
Reply from 10.10.10.3: bytes=32 time=52ms TTL=64
Reply from 10.10.10.3: bytes=32 time<1ms TTL=64
Reply from 10.10.10.3: bytes=32 time<1ms TTL=64
Reply from 10.10.10.3: bytes=32 time<1ms TTL=64

Ping statistics for 10.10.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 52ms, Average = 13ms

C:\Users\mkign>ping 10.10.10.4

Pinging 10.10.10.4 with 32 bytes of data:
Reply from 10.10.10.4: bytes=32 time<1ms TTL=64
Reply from 10.10.10.4: bytes=32 time<1ms TTL=64
Reply from 10.10.10.4: bytes=32 time=2ms TTL=64
Reply from 10.10.10.4: bytes=32 time<1ms TTL=64

Ping statistics for 10.10.10.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\Users\mkign>
```

## 5. CONCLUSION

packet sniffer is not just a hacker's tool. It can be used for network traffic monitoring, traffic analysis, troubleshooting and other useful purposes. Packet sniffers can capture things like passwords and usernames or other sensitive information. Networks Sniffing in non switched network is easy but sniffing in switched network is difficult because we use switches in network which narrow the traffic and send to particular system, so for sniffing in this type of network we use some methods. There are many available tools. Packet sniffer can be enhanced in future by Incorporating features like making the packet sniffer program platform independent, and making tool by neural network. 10 GBPS LAN which are used currently, sniffing can done on this rate in future very effectively.

## 6. REFERENCES

- [Tkinter tkFileDialog module - Python Tutorial](#)
- [Python GUI Development with Tkinter: Part 3](#)
- [https://stackoverflow.com/](#)
- [https://www.datacamp.com/community/tutorials/gui-tkinter-python](#)

## 7. ANNEXURE

### 7.1 Coding

**nps.py**

'''

**Sniff packets and return a list of packets.**

**Arguments:**

**count:** number of packets to capture. 0 means infinity.

**store:** whether to store sniffed packets or discard them

**prn:** function to apply to each packet. If something is returned, it

is displayed.

**Ex:** prn = lambda x: x.summary()

**filter:** BPF filter to apply.

**lfilter:** Python function applied to each packet to determine if

further action may be done.

**Ex:** lfilter = lambda x: x.haslayer(Padding)

**offline:** PCAP file (or list of PCAP files) to read packets from,

instead of sniffing them

**timeout:** stop sniffing after a given time (default: None).

**L2socket:** use the provided L2socket (default: use conf.L2listen).

**opened\_socket:** provide an object (or a list of objects) ready to use

`.recv()` on.

**stop\_filter:** Python function applied to each packet to determine if we have to stop the capture after this packet.

**Ex:** `stop_filter = lambda x: x.haslayer(TCP)`

**iface:** interface or list of interfaces (default: None for sniffing on all interfaces).

The `iface`, `offline` and `opened_socket` parameters can be either an element, a list of elements, or a dict object mapping an element to a label (see examples below).

**Examples:**

```
>>> sniff(filter="arp")
```

```
>>> sniff(lfilter=lambda pkt: ARP in pkt)
```

```
>>> sniff(iface="enp0s8", prn=Packet.summary)
```

```
>>> sniff(iface=["enp0s8", "mon0"],
```

```
...     prn=lambda pkt: "%s: %s" % (pkt.sniffed_on,
```

```
...                               pkt.summary()))
```

```
>>> sniff(iface={"enp0s8": "Ethernet", "mon0": "Wifi"},
```

```
...     prn=lambda pkt: "%s: %s" % (pkt.sniffed_on,
```

```
...                               pkt.summary()))
```

```
'''
```

**#Importing the necessary modules**

**import logging**

**from datetime import datetime**

**import subprocess**

**import sys**

**from tkinter import \***

**import tkinter.ttk as ttk**

**from tkinter.ttk import Notebook**

**#This will suppress all messages that have a lower level of seriousness than error messages,  
while running or loading Scapy**

**logging.getLogger("scapy.runtime").setLevel(logging.ERROR)**

**logging.getLogger("scapy.interactive").setLevel(logging.ERROR)**

**logging.getLogger("scapy.loading").setLevel(logging.ERROR)**

**try:**

**from scapy.all import \***

**except ImportError:**

**print("Scapy package for Python is not installed on your system.")**

**sys.exit()**



**#Printing a message to the user; always use "sudo scapy" in Linux!**

```
print("\n! Make sure to run this program as ROOT !\n")
```

**#Asking the user for some parameters: interface on which to sniff, the number of packets to sniff, the time interval to sniff, the protocol**

**#Asking the user for input - the interface on which to run the sniffer**

```
net_iface = input("* Enter the interface on which to run the sniffer (e.g. 'enp0s8'): ")
```

**#Setting network interface in promiscuous mode**

**'''**

**In computer networking, promiscuous mode or "promisc mode"[1] is a mode for a wired network interface controller (NIC) or wireless network interface controller (WNIC) that causes the controller to pass all traffic it receives to the central processing unit (CPU) rather than passing only the frames that the controller is intended to receive.**

**This mode is normally used for packet sniffing that takes place on a router or on a computer connected to a hub.**

**'''**

**try:**

```
subprocess.call(["ifconfig", net_iface, "promisc"], stdout = None, stderr = None, shell = False)
```

**except:**

```
print("\nWARNING! Failed to configure interface as promiscuous.\n")
```

**else:**

**#Executed if the try clause does not raise an exception**

**print("\n=>Interface %s was set to PROMISC mode!\n" % net\_iface)**

**#Asking the user for the number of packets to sniff (the "count" parameter)**

**pkt\_to\_sniff = input("\n\* Enter the number of packets to capture (0 is infinity): ")**

**#Considering the case when the user enters 0 (infinity)**

**if int(pkt\_to\_sniff) != 0:**

**print("\n=>The program will capture %d packets!\n" % int(pkt\_to\_sniff))**

**elif int(pkt\_to\_sniff) == 0:**

**print("\n=>The program will capture packets until the timeout!\n")**

**#Asking the user for the time interval to sniff (the "timeout" parameter)**

**time\_to\_sniff = input("\n\* Enter the number of seconds to run the capture: ")**

**#Handling the value entered by the user**

**if int(time\_to\_sniff) != 0:**

**print("\n=>The program will capture packets for %d seconds!\n" % int(time\_to\_sniff))**

**#Asking the user for any protocol filter he might want to apply to the sniffing process**

**#For this example I chose three protocols: ARP, ICMP**

**#You can customize this to add your own desired protocols**

**proto\_sniff = input("\n\* Enter the protocol to filter by (arp || icmp || 0 is all): ")**

**#Considering the case when the user enters 0 (meaning all protocols)**

**if (proto\_sniff == "arp") or (proto\_sniff == "icmp"):**

**print("\n=>The program will capture only %s packets!\n" % proto\_sniff.upper())**

**elif (proto\_sniff) == "0":**

**print("\n=>The program will capture all protocols!\n")**

**#Asking the user to enter the name and path of the log file to be created**

**file\_name = input("\n\* Please give a name to the log file: ")**

**#Creating the text file (if it doesn't exist) for packet logging and/or opening it for appending**

**sniffer\_log = open(file\_name, "a")**

**#This is the function that will be called for each captured packet**

**#The function will extract parameters from the packet and then log each packet to the log file**

**def packet\_log(packet):**

**#Getting the current timestamp**

**now = datetime.now()**

**#Writing the packet information to the log file, also considering the protocol or 0 for all protocols**

**if proto\_sniff == "0":**

**#Writing the data to the log file**

**print("Time: " + str(now) + " Protocol: ALL" + " SMAC: " + packet[0].src + " DMAC: " + packet[0].dst, file = sniffer\_log)**

**elif (proto\_sniff == "arp") or (proto\_sniff == "icmp"):**

**#Writing the data to the log file**

**print("Time: " + str(now) + " Protocol: " + proto\_sniff.upper() + " SMAC: " + packet[0].src + " DMAC: " + packet[0].dst, file = sniffer\_log)**

**#Printing an informational message to the screen**

```
print("\n* Starting the capture...")
```

**#Running the sniffing process (with or without a filter)**

```
if proto_sniff == "0":
```

```
    sniff(iface = net_iface, count = int(pkt_to_sniff), timeout = int(time_to_sniff), prn =  
    packet_log)
```

```
elif (proto_sniff == "arp") or (proto_sniff == "icmp"):
```

```
    sniff(iface = net_iface, filter = proto_sniff, count = int(pkt_to_sniff), timeout =  
    int(time_to_sniff), prn = packet_log)
```

```
else:
```

```
    print("\nWARNING! Could not identify the protocol.\n")
```

```
    sys.exit()
```

**#Printing the closing message**

```
print("\n* Displaying the captured packets saved in %s.\n" % file_name)
```

**#Closing the log file**

```
sniffer_log.close()
```

```
#subprocess.call(" python3 disp.py 1", shell=True)
```

```
#Displaying the captured packets information through GUI mode
```

```
container=Tk()
```

```
container.title('Network Packet Sniffer')
```

```
container.geometry('800x600')
```

```
generalites=Frame(container,bg='powder blue')
```

```
generalites.pack(side=BOTTOM)
```

```
s_generalites= Scrollbar(generalites)
```

```
T_generalites= Text(generalites,bg='powder blue',width=350,height=350)
```

```
s_generalites.pack(side=RIGHT, fill=Y)
```

```
T_generalites.pack(side=LEFT, fill=Y)
```

```
s_generalites.config(command=T_generalites.yview)
```

```
T_generalites.config(yscrollcommand=s_generalites.set)
```

```
filename =(file_name)
```

```
fichier = open(filename, "r")
```

```
content_generalites= fichier.read()
```

```
fichier.close()
```

```
T_generalites.insert(END, content_generalites)
```

```
container.mainloop()
```

```
print("Thank you for using the service!")
```

```
#End of the program.
```

## 7.2 INTERFACE

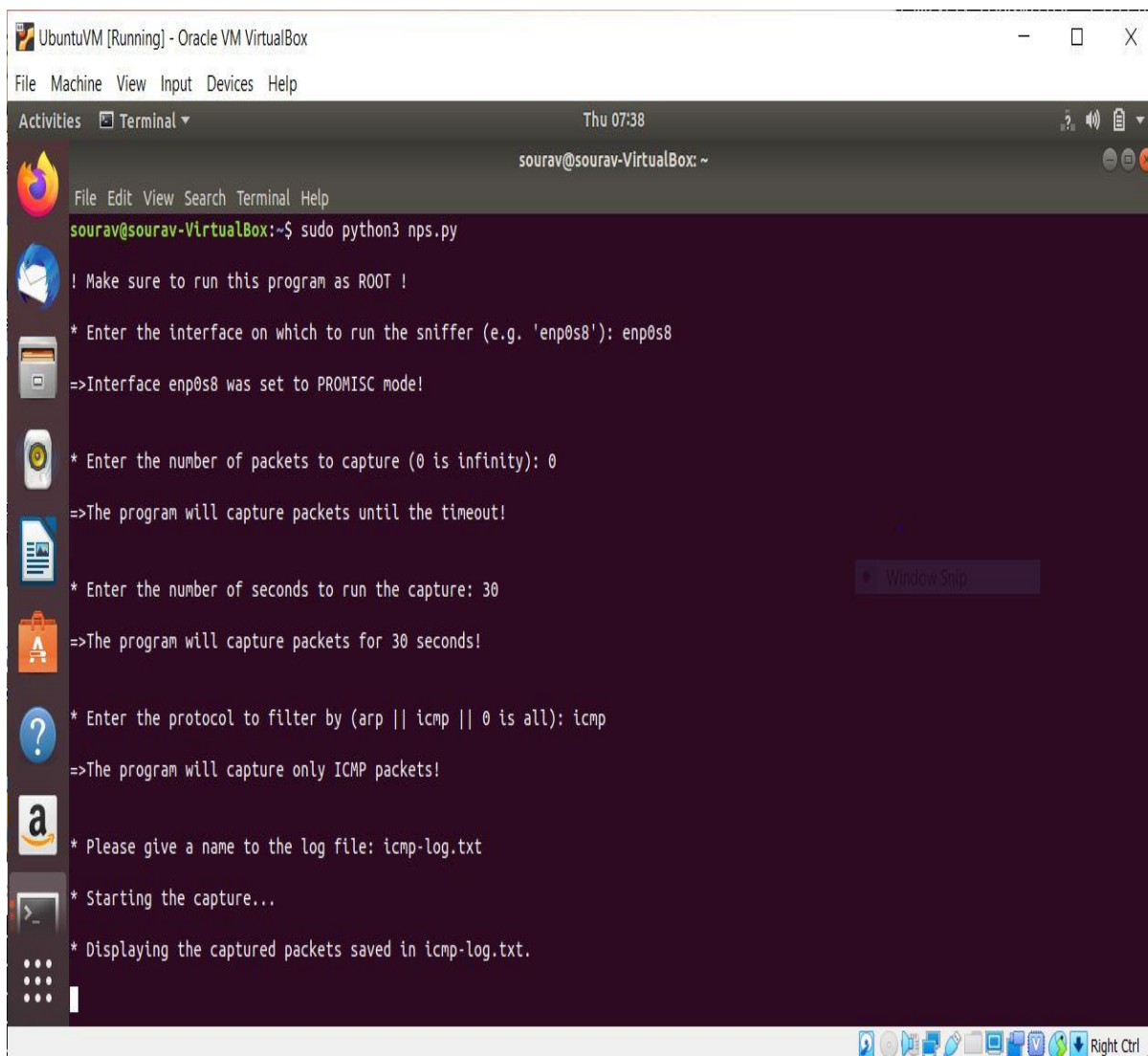
### I. (CLI) Command line Interface

Asking the User for Input, extracting parameters from Packets,  
Writing to a Log File by running the Sniffer in CLI mode.

### II. GUI(Graphical User Interface)

Displaying the log file in the GUI interface mode

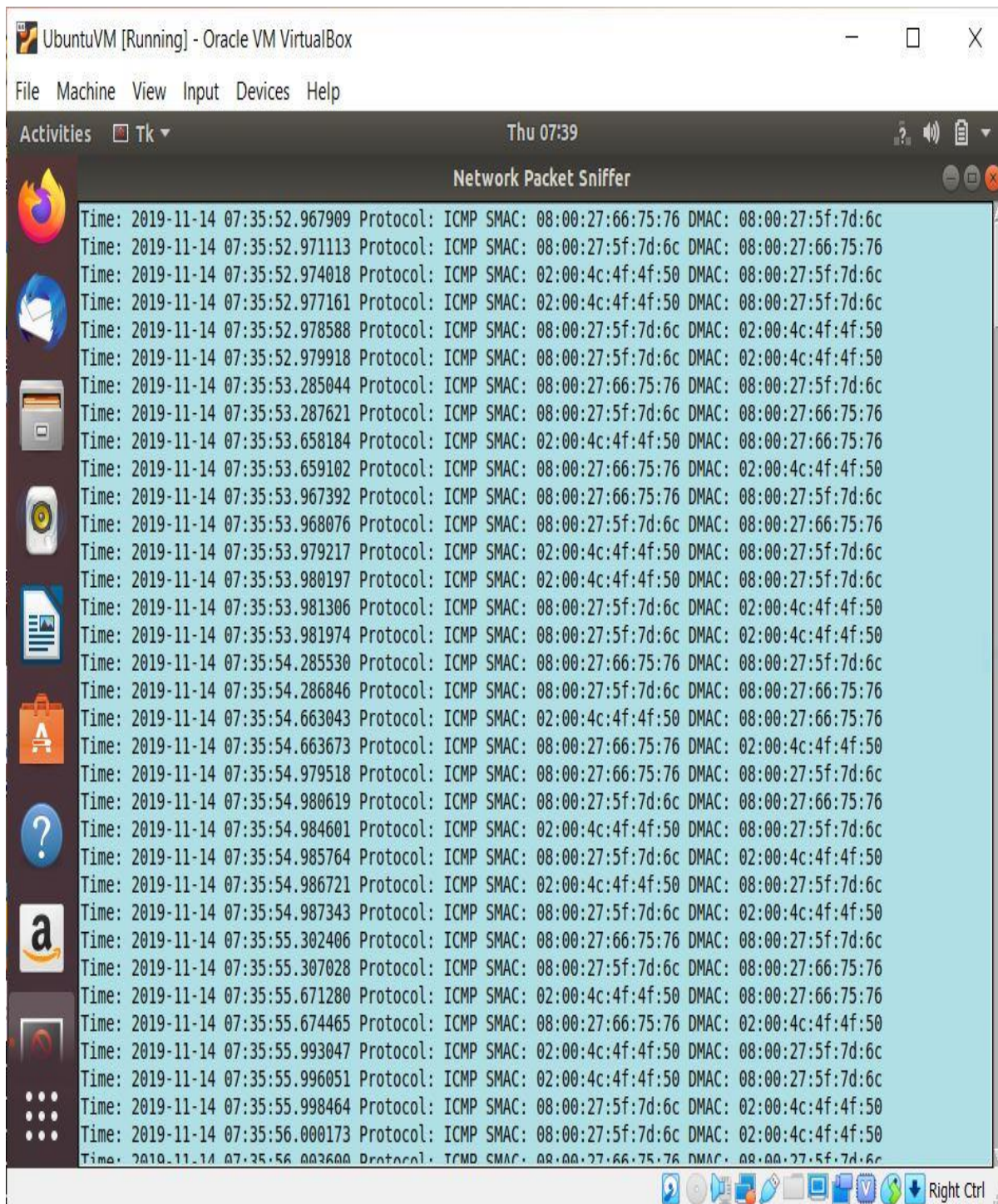
#### ❖ Capturing ICMP protocol packets



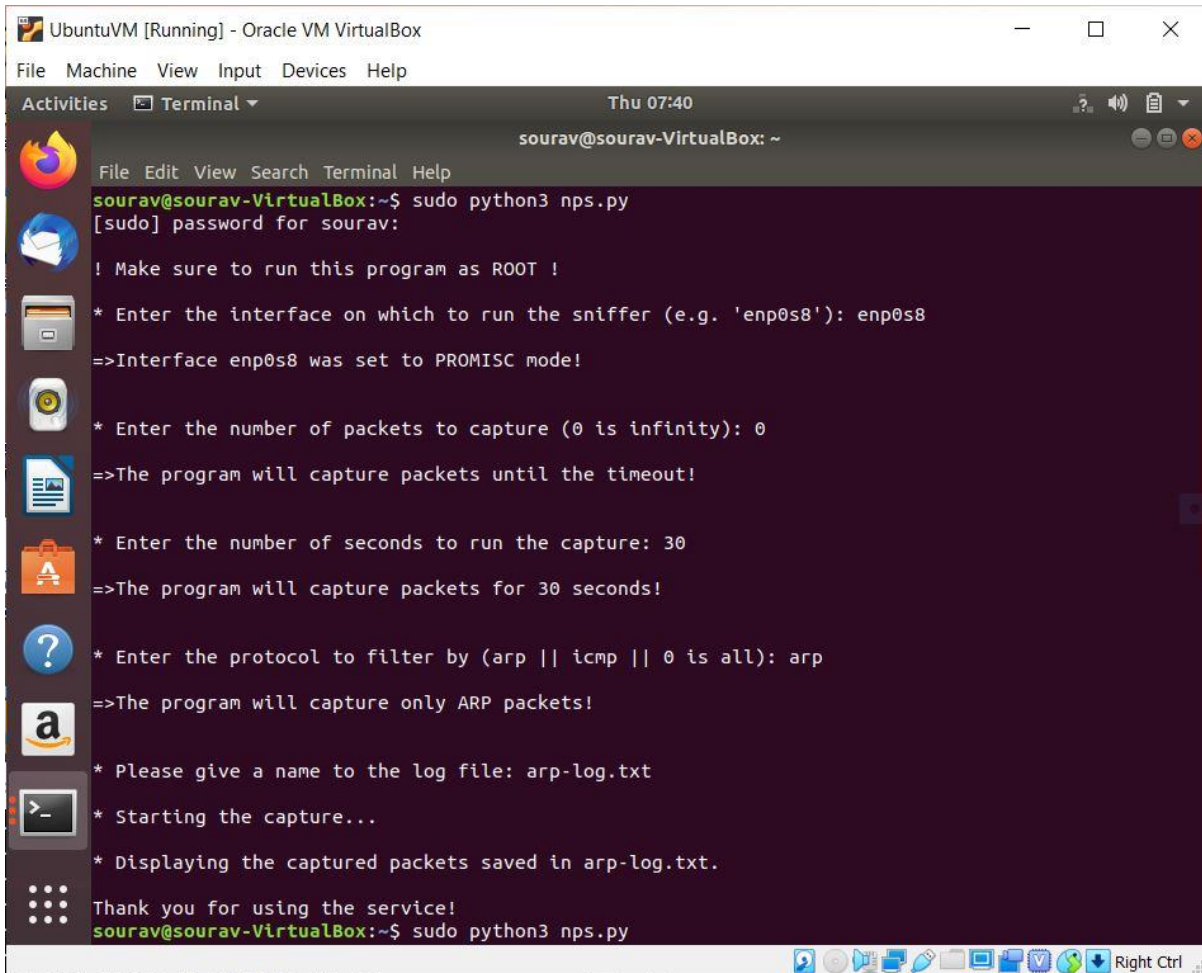
```
UbuntuVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thu 07:38
sourav@sourav-VirtualBox: ~
File Edit View Search Terminal Help
sourav@sourav-VirtualBox:~$ sudo python3 nps.py
! Make sure to run this program as ROOT !
* Enter the interface on which to run the sniffer (e.g. 'enp0s8'): enp0s8
=>Interface enp0s8 was set to PROMISC mode!
* Enter the number of packets to capture (0 is infinity): 0
=>The program will capture packets until the timeout!
* Enter the number of seconds to run the capture: 30
=>The program will capture packets for 30 seconds!
* Enter the protocol to filter by (arp || icmp || 0 is all): icmp
=>The program will capture only ICMP packets!
* Please give a name to the log file: icmp-log.txt
* Starting the capture...
* Displaying the captured packets saved in icmp-log.txt.
```



## ❖ Output for captured ICMP protocol packets



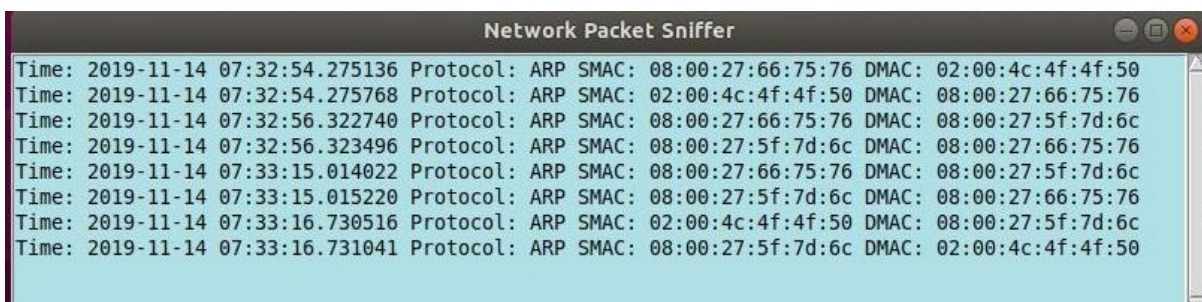
## ❖ Capturing ARP protocol packets



The screenshot shows a terminal window titled "UbuntuVM [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
sourav@sourav-VirtualBox: ~  
sourav@sourav-VirtualBox:~$ sudo python3 nps.py  
[sudo] password for sourav:  
! Make sure to run this program as ROOT !  
* Enter the interface on which to run the sniffer (e.g. 'enp0s8'): enp0s8  
=>Interface enp0s8 was set to PROMISC mode!  
* Enter the number of packets to capture (0 is infinity): 0  
=>The program will capture packets until the timeout!  
* Enter the number of seconds to run the capture: 30  
=>The program will capture packets for 30 seconds!  
* Enter the protocol to filter by (arp || icmp || 0 is all): arp  
=>The program will capture only ARP packets!  
* Please give a name to the log file: arp-log.txt  
* Starting the capture...  
* Displaying the captured packets saved in arp-log.txt.  
Thank you for using the service!  
sourav@sourav-VirtualBox:~$ sudo python3 nps.py
```

## ❖ Output for captured ARP protocol packets

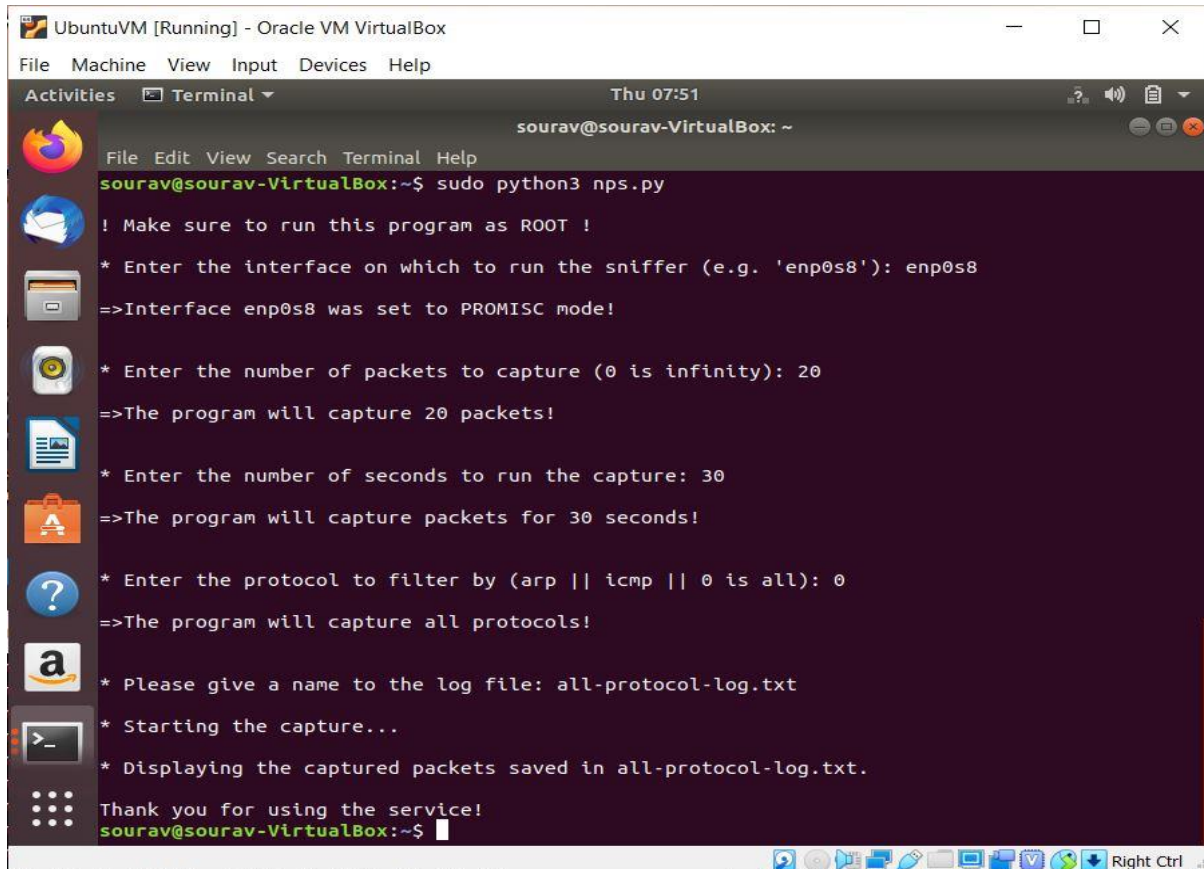


The screenshot shows a window titled "Network Packet Sniffer" displaying the following captured packets:

Time	Protocol	SMAC	DMAC
2019-11-14 07:32:54.275136	ARP	08:00:27:66:75:76	02:00:4c:4f:4f:50
2019-11-14 07:32:54.275768	ARP	02:00:4c:4f:4f:50	08:00:27:66:75:76
2019-11-14 07:32:56.322740	ARP	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:32:56.323496	ARP	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:33:15.014022	ARP	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:33:15.015220	ARP	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:33:16.730516	ARP	02:00:4c:4f:4f:50	08:00:27:5f:7d:6c
2019-11-14 07:33:16.731041	ARP	08:00:27:5f:7d:6c	02:00:4c:4f:4f:50



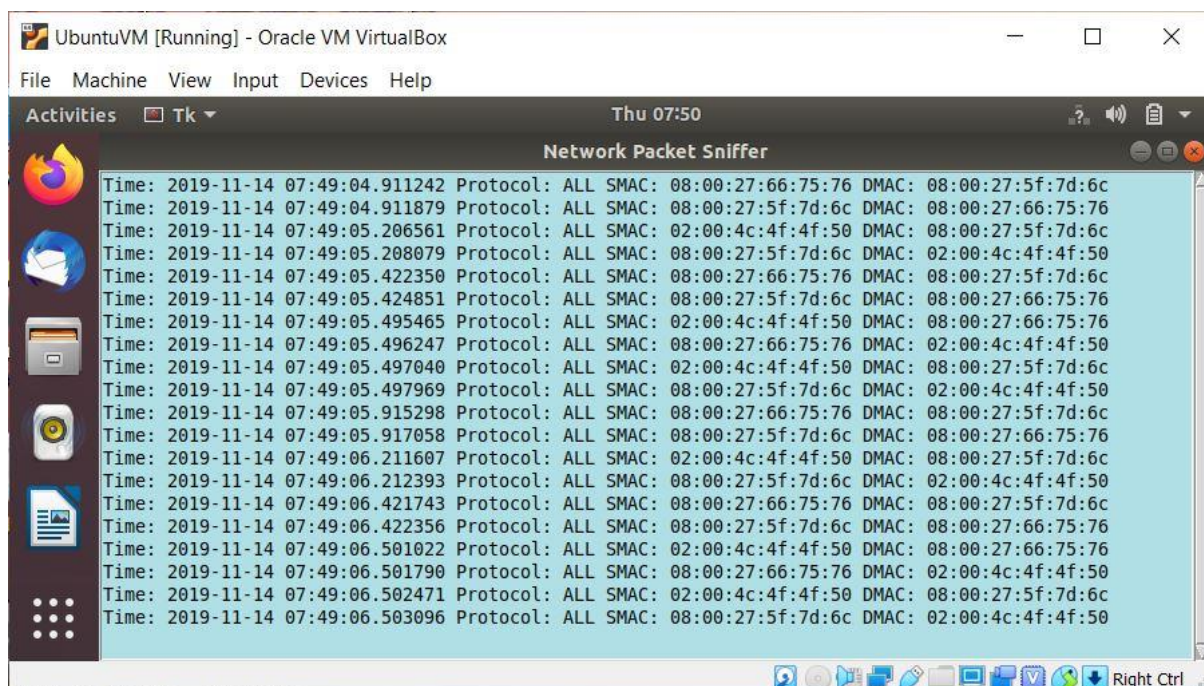
## ❖ Capturing All protocol packets



The screenshot shows a terminal window titled "UbuntuVM [Running] - Oracle VM VirtualBox". The user is logged in as "sourav@sourav-VirtualBox: ~". The terminal displays the output of the command `sudo python3 nps.py`. The program prompts the user for several inputs: the interface (enp0s8), the number of packets to capture (20), the number of seconds to run the capture (30), the protocol to filter by (0 for all), and a name for the log file (all-protocol-log.txt). The program then starts the capture and displays the captured packets saved in the log file.

```
sourav@sourav-VirtualBox: ~  
File Edit View Search Terminal Help  
sourav@sourav-VirtualBox:~$ sudo python3 nps.py  
! Make sure to run this program as ROOT !  
* Enter the interface on which to run the sniffer (e.g. 'enp0s8'): enp0s8  
=>Interface enp0s8 was set to PROMISC mode!  
* Enter the number of packets to capture (0 is infinity): 20  
=>The program will capture 20 packets!  
* Enter the number of seconds to run the capture: 30  
=>The program will capture packets for 30 seconds!  
* Enter the protocol to filter by (arp || icmp || 0 is all): 0  
=>The program will capture all protocols!  
* Please give a name to the log file: all-protocol-log.txt  
* Starting the capture...  
* Displaying the captured packets saved in all-protocol-log.txt.  
Thank you for using the service!  
sourav@sourav-VirtualBox:~$
```

## ❖ Output for captured All protocol packets



The screenshot shows a window titled "Network Packet Sniffer" displaying a list of captured packets. Each line represents a packet capture with the following fields: Time, Protocol, SMAC, and DMAC. The data is as follows:

Time	Protocol	SMAC	DMAC
2019-11-14 07:49:04.911242	ALL	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:49:04.911879	ALL	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:49:05.206561	ALL	02:00:4c:4f:4f:50	08:00:27:5f:7d:6c
2019-11-14 07:49:05.208079	ALL	08:00:27:5f:7d:6c	02:00:4c:4f:4f:50
2019-11-14 07:49:05.422350	ALL	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:49:05.424851	ALL	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:49:05.495465	ALL	02:00:4c:4f:4f:50	08:00:27:66:75:76
2019-11-14 07:49:05.496247	ALL	08:00:27:66:75:76	02:00:4c:4f:4f:50
2019-11-14 07:49:05.497040	ALL	02:00:4c:4f:4f:50	08:00:27:5f:7d:6c
2019-11-14 07:49:05.497969	ALL	08:00:27:5f:7d:6c	02:00:4c:4f:4f:50
2019-11-14 07:49:05.915298	ALL	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:49:05.917058	ALL	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:49:06.211607	ALL	02:00:4c:4f:4f:50	08:00:27:5f:7d:6c
2019-11-14 07:49:06.212393	ALL	08:00:27:5f:7d:6c	02:00:4c:4f:4f:50
2019-11-14 07:49:06.421743	ALL	08:00:27:66:75:76	08:00:27:5f:7d:6c
2019-11-14 07:49:06.422356	ALL	08:00:27:5f:7d:6c	08:00:27:66:75:76
2019-11-14 07:49:06.501022	ALL	02:00:4c:4f:4f:50	08:00:27:66:75:76
2019-11-14 07:49:06.501790	ALL	08:00:27:66:75:76	02:00:4c:4f:4f:50
2019-11-14 07:49:06.502471	ALL	02:00:4c:4f:4f:50	08:00:27:5f:7d:6c
2019-11-14 07:49:06.503096	ALL	08:00:27:5f:7d:6c	02:00:4c:4f:4f:50