

**New IPS Based On Modified Forks Of Wireshark  
And Snort Merged Into A Single Product.**

**MINT 709: Capstone Project Report**

**ARJUN JOSHI**

**Supervisor: Mr. Lenonard Rogers**



## **ABSTRACT**

---

Intrusion prevention has become an extremely important feature of the Defense-in-Depth strategy. The intention of this project is to build the Linux-based open source system which functions as an IPS based on a merger/rewrite of Wireshark and Snort with IP Geolocation built in. This system should work like a wire in the network without causing much delay. Wireshark will capture and analyze every packet. It will generate basic reports for further analysis.

## **ACKNOWLEDGEMENT**

---

I express my cavernous sense of obligation and gratitude to my supervisor Mr. Leonard Rogers for his genuine guidance and constant encouragement throughout this project work. I am highly obliged as my honourable supervisor has devoted his valuable time and shared his expertise knowledge.

I pay my profound gratefulness to Dr. Michael Macgregor for giving me an opportunity to carry out the project work. I must thank him for sparing his valuable time from his busy schedule.

I extend my sincere thanks to Mr. Shahnawaz Mir for providing me such an opportunity to do my project work.

I also wish to express my heartfelt appreciation to my friends, colleagues and many who have rendered their support for the successful completion of the project, both explicitly and implicitly.

ARJUN JOSHI

# INDEX

---

<b>Abstract</b>	I
<b>Acknowledgement</b>	II
<b>List Of Figures</b>	IV
<b>List Of Tables</b>	V
<b>1. INTRODUCTION</b>	1
1.1 Description of the problem	1
1.2 Organization of Report	2
<b>2. COMPONENTS</b>	3
2.1 Linux-Open Source	3
2.2 Ubuntu	3
2.3 Virtualbox/Virtuallization	3
2.4 Intrusion Prevention System (IPS) and SNORT	4
2.5 Wireshark	5
2.6 IPTABLES FIREWALL	6
2.6.1 TTL Target	6
2.6.2 Proxy ARP	7
<b>3. Test Procedure</b>	8
3.1 Network Diagram	8
3.2 Configuration Script	8
<b>4. RESULTS</b>	11
<b>5. CONCLUSION</b>	19
<b>REFERENCES</b>	20

---

## LIST OF FIGURES

---

No	Title	Page
1.1.1	Project Block Diagram	1
1.1.2	System as a Wire	1
3.1.1	Network Diagram	8
4.1	SELinux Disabled	11
4.2	Ping from VBOX1 to VBOX3	11
4.3	Mirror packet on VBOX4 Management Interface	12
4.4	TCPDUMP and Wireshark capture at VBOX3	13
4.5	TCPDUMP and Wireshark capture at VBOX4	14
4.6	GEOIP Reports Generation	15
4.7.1	Hping3 for Generation of flood traffic	16
4.7.2	Top Talkers	16
4.7.3	Packet Counts by Protocols	17
4.7.4	Packet Length Statistics	17
4.7.5	Packet Length Statistics by Protocol	18

---

## LIST OF TABLES

---

No.	Title	Page
2.6.1.1	TTL target	6

# CHAPTER 1

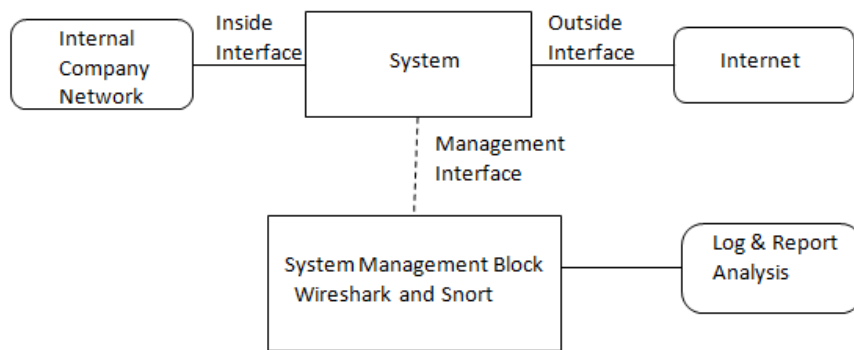
---

## INTRODUCTION

This Chapter explains description of the problem with a Project Block Diagram. Organization of report is also carried out in section 1.2.

### 1.1 Description of the problem

This is a multi-phase project with several phases. I have dealt with the initial phase problem: “Interface modification and packet collection/replication”



*Figure 1.1.1: Project Block Diagram*



*Figure 1.1.2: System as a Wire*

This initial phase is to create a Linux operating system that can monitor and record information of all traffic passing through two of its interfaces. One of the main functions in this new system is to pass packets between the inside and outside interfaces without causing delay between the interfaces. Figure 1.1.1 Shows block diagram of Project. In other words, as far as the packet is concerned it is just traveling on the wire without crossing any interfaces as shown in above Figure 1.1.2. To do this, it means that the packet’s hop count is not decremented as it passes between interfaces.

While this does bend/violate some of the rules around the standards of the TCP/IP, the reasons for doing this is to prevent the IPS from being easily detected, and to ensure that trace backs of the communications stream are not seen as going through an additional hop. Also, the inside and outside interfaces do not require an IP Address to function nor do they use their own MAC address.

In the initial phase, they are simply passing traffic back and forth to each other and to the third interface. This means that any inbound traffic received at either the inside or outside interfaces are fully passed to their counterpart and the management interface while ensuring that any data that would normally be changed while going through a hop is not changed and that the packets integrity is maintained.

The management interface does have an IP Address defined and is where the IPS (Wireshark/Snort) solutions are built. As the traffic is replicated to this interface, the Wireshark function (Phase One) is used to capture and analyze every packet. This includes checking the packet's IP Geolocation address from the Geolocation database.

Traffic analysis such as Top 10 or Top 100 talkers on the network can be tracked. Traffic types can also be tracked to better manage QOS loads, allowing for better and even automated traffic shaping, based on the weighting, type and network load of protocols travelling on the network.

### **1.2 Organization of Report**

This report consists of 5 chapters in total. The framework for the report is described as follows:

Chapter 1 provides a brief introduction about initial phase of this project with explanation of project diagram. Components of this project are described in the Chapter 2. Chapter 3 discusses the Test Procedure including network diagram and configuration script which fulfills the initial phase requirement. Result of this project is discussed in Chapter 4. Finally Chapter 5 gives the conclusive remarks of work.



## CHAPTER 2

---

### COMPONENTS

In this chapter components are described. It gives the basic description and idea of every tool used in this project.

#### 2.1 Linux-Open Source

Linux is the widely-used open source operating system. As an operating system, Linux is software that resides below all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware. Ref. [1]

#### 2.2 Ubuntu

Ubuntu is basically a debian-based Linux OS. Ubuntu Touch edition is used for personal computers, Smartphone and tablets. Ubuntu server edition runs network servers. I have used Ubuntu version 14.04.1 for this project. Ref. [2]

#### 2.3 Virtual box/Virtualization

It is a kind of hypervisor used to run an OS in a special environment, which is called a virtual machine. There is continuous development & additional features are implemented inside virtual box. It contains a GUI interface (Qt), as well as headless and command-line tools (SDL) to control and run virtual machines. I have used Virtual Box Version 5.1.4. Ref. [3, 4]

There are 8 different virtual PCI Ethernet cards in VM. For each such card, there is an option to select

1. The hardware which will be virtualized
2. The mode of virtualization that the virtual card can be operating in with respect to physical networking hardware on the host.

#### Networking Modes

Networking adapters can be separately configured to work in one of the following modes:

##### Network Address Translation (NAT)

This mode is used to surf the Web, view e-mail and download files. The "NAT network" is a new NAT scheme introduced in Virtual Box 4.3.

### **Not attached**

This mode indicates network card exists but is not connected, as if no Ethernet cable is plugged into the card.

### **Bridged networking**

There is a direct connection between virtual Box and the installed network cards. Virtual Box exchanges network packets directly, bypassing host OS network stack.

### **Internal networking**

Various kind of software-based network can be created through this mode which is visible to selected virtual machines.

### **Host-only networking**

It creates a separate network that contains the host and a set of VMs, without the need of host's physical network interface. A separate virtual network interface is created on top of the host that provides connection among VMs and the host.

### **Generic networking**

This mode shares the same generic network interface. It allows the user to pick up a driver that can be included with Virtual Box or be distributed in another extension pack.

## **2.4 Intrusion Prevention System and SNORT**

IPS is a threat prevention technology which verifies network traffic flows to discover and forestall vulnerability exploits. Vulnerability exploits occur within the kind of malicious inputs to a target application or service that attackers use to interrupt and gain management of an application or machine. After successful exploit, the aggressor will disable the target application (resulting in a denial-of-service state), or will probably gain access to all or any rights and permissions available to the compromised application.

**Intrusion prevention systems** track network traffic for malicious activity. The important functions of intrusion prevention systems are to spot malicious activity, log info concerning this activity, report it and block or stop it. Ref. [5]

### **Categorization**

Intrusion prevention systems may be categorized into four totally different types:

1. **Host-based intrusion prevention system:** It tracks one host for malicious activity by analyzing events occurring inside particular host using an installed software package.
2. **Wireless intrusion prevention systems:** It tracks a wireless network for malicious traffic as per the behavior of wireless networking protocols.

3. **Network-based intrusion prevention system:** It tracks the complete network for malicious traffic by checking protocol activity.
4. **Network behavior analysis:** It examines network traffic to spot threats that generate uncommon traffic flows, like distributed denial of service (DDoS) attacks, certain varieties of malware and policy violations.

### Detection Techniques

There are three detection techniques for intrusion prevention systems: stateful protocol analysis, statistical anomaly-based, and signature-based Ref. [5]

1. **Stateful Protocol Analysis based:** It identifies deviations of protocol states. It compares observed events with respect to “predetermine profiles of normally accepted definitions of benign activity.
2. **Statistical anomaly-based:** It monitors network traffic and based on traffic compares against an established baseline. What is "normal" for that network will be identified by the baseline.
3. **Signature-Based:** It monitors packets within the Network and compares it with pre-determined and pre-configured attack patterns known as signatures.

### Snort:

It is an open source and free network detection system and prevention system which was created by Martin Roesch in 1998. It has the capability to perform live traffic analysis and packet logging on IP networks.

It performs content searching, matching and protocol analysis. The program may also be accustomed to notice probes, attacks, including common gateway interface, operating system fingerprinting attempts, server message block probes, stealth port scans, and buffer overflows.

There are three basic modes of configuration for snort: packet logger, sniffer and network intrusion detection. The program will log packet to the disk in packet logger mode. The program will read and display network packets on the console in sniffer mode. The program will monitor and analyze network traffic with respect to rule defined by the user in intrusion detection mode. Ref. [6]

## 2.5 Wireshark

It is an open source and free packet analyzer. It is normally used for network analysis, troubleshooting, education and software & communications protocol development.

It is using pcap to capture packets. It also has a terminal based version called TShark. User can see all traffic visible on interface using promiscuous mode. Wireless network interface controllers can also be kept into monitor mode. Ref. [7]

## 2.6 IPTABLES Firewall

Most of Linux distributions contain IPtables which is a standard firewall. It works by matching every packet that crosses the networking interface against a group of rules to come to a decision what to try and do.

There are several choices to determine which packets match a particular rule. You can match the source or destination address or port, the packet protocol type, the interface that is being used, etc.

When the outlined pattern matches, the action that takes place is named a **target**. A target is often being a final policy decision for the packet, like drop, or accept. It can also be move the packet to a distinct chain for process, or just log the encounter.

These rules are organized into teams referred to as **chains**. A chain could be a set of rules that a packet is checked against consecutively. Once the packet matches one of the rules, it executes the associated task and isn't checked against the rest of rules in the chain. Iptables can also keep track of connections.

There are three main chains. They are:

- **INPUT**: It controls all packets that are addressed to your server.
- **OUTPUT**: It keeps rules for traffic created by your server.
- **FORWARD**: It deals with traffic destined for different servers that are not created on your server. This chain is normally a way to configure your server to route requests to different machines. Ref. [8]

### 2.6.1 TTL Target

It modifies the Time To Live field in IP header. The **TTL** target is applied within the mangle table. There are 3 ways to define TTL Target as defined in the table below: Ref. [9]

*Table 2.6.1.1 TTL target*

1	<b>--ttl-set</b>
Ex	iptables -t mangle -A PREROUTING -i eth0(interface) -j TTL --ttl-set 64
Detail	The <b>--ttl-set</b> option specifies the <b>TTL</b> target which TTL value to keep on the packet in question.
2	<b>--ttl-dec</b>
Ex	iptables -t mangle -A PREROUTING -i eth0 (interface) -j TTL --ttl-dec 1
Details	The <b>--ttl-dec</b> option specifies the <b>TTL</b> target to decrement Time To Live value by the amount specified after the <b>--ttl-dec</b> option.

	Ex, if the TTL for an incoming packet was 53 & we had set <b>--ttl-dec 3</b> , the packet would leave host with a TTL value of 49.
3	<b>--ttl-inc</b>
Ex	iptables -t mangle -A PREROUTING -i eth0 (interface) -j TTL --ttl-inc 1
Detail	The <b>--ttl-inc</b> option specifies the <b>TTL</b> target to increment Time To Live value with the value specified to the <b>--ttl-inc</b> option. Ex, A packet entering with a TTL of 53 would leave the host with TTL 56, if we have specified <b>--ttl-inc 4</b> .

### 2.6.2 Proxy ARP

A network switch is a kind of bridge with many ports. Proxy ARP can publish ARP entries on 1 interface for machines on 2nd interface. Here, machines on one network segment can reach machines on other segment via intermediary machine. Ref. [10]

#### Example:

Enable IP forwarding:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Proxy ARP:

```
echo 1 > /proc/sys/net/ipv4/conf/eth0 (interface)/proxy_arp
```

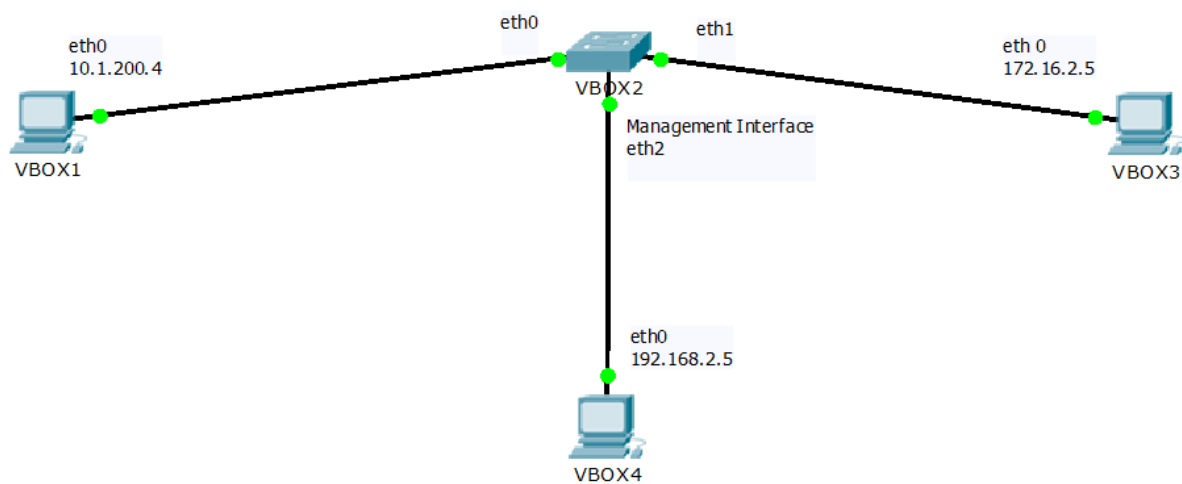
```
echo 1 > /proc/sys/net/ipv4/conf/eth1 (interface)/proxy_arp
```

## CHAPTER 3

### TEST PROCEDURE

This Chapter gives a detailed description about how testing is carried out for this Project. Section 3.1 shows network diagram, while section 3.2 describes configuration script.

#### 3.1 Network Diagram



*Figure 3.1.1: Network Diagram*

- As per the diagram, I am successfully able to ping VBOX3 from VBOX1
- TTL value is also not affecting after changes in VBOX2 so VBOX 2 is behaving as a wire
- I am also able to mirror VBOX1 ---> VBOX3 traffic on VBOX4 which will act as a management box
- IP address has been removed from the interfaces of VBOX2. There is only MAC addresses assigned on interfaces for VBOX2.

#### 3.2 Configuration Script

##### Part 1: IP assignment on interface

```
VBOX1:
Ifconfig eth0 10.1.200.4 netmask 255.255.255.0
VBOX3:
Ifconfig eth0 172.16.2.5 netmask 255.255.255.0
VBOX4:
Ifconfig eth0 192.168.2.5 netmask 255.255.255.0
```

### Part 2: Static Route Assignment on every VBOX

VBOX1:  
IP route add 172.16.2.0/24 dev eth0  
VBOX2:  
IP route add 172.16.2.0/24 dev eth1  
IP route add 10.1.200.0/24 dev eth0  
IP route add 192.168.2.0/24 dev eth2  
VBOX3:  
IP route add 10.1.200.0/24 dev eth0

### Part 3: Forward interface traffic via VBOX2 and Proxy ARP configuration

VBOX2:  
Sysctl -w net.ipv4.ip\_forward=1  
Iptables -A FORWARD -i eth0 -j ACCEPT  
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy\_arp  
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy\_arp  
echo 1 > /proc/sys/net/ipv4/conf/eth2/proxy\_arp

### Part 4: VBOX2 Act as a Wire/Hop count unaffected by VBOX2

VBOX2:  
Iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1  
Iptables -t mangle -A PREROUTING -i eth1 -j TTL --ttl-inc 1

### Part 5: Mirror Traffic to VBOX4 Management Interface

Iptables -t mangles -A PREROUTING -j TEE --gateway 192.168.2.5

### Following commands are used for Permanent configuration:

VBOX1:  
gedit /etc/network/interfaces  
auto eth0  
    iface eth0 inet static  
        address 10.1.200.4  
        netmask 255.255.255.0  
up route add -net 10.1.200.0/24 dev eth0  
up route add -net 172.16.2.0/24 dev eth0

VBOX3:  
gedit /etc/network/interfaces  
auto eth0  
    iface eth0 inet static  
        address 172.16.2.5  
        netmask 255.255.255.0  
up route add -net 10.1.200.0/24 dev eth0  
up route add -net 172.16.2.0/24 dev eth0

VBOX4:

```
gedit /etc/network/interfaces
auto eth0
    iface eth0 inet static
        address 192.168.2.5
        netmask 255.255.255.0
up route add -net 192.168.2.0/24 dev eth0
```

VBOX2:

```
gedit /etc/network/interfaces
```

```
auto eth0
iface eth0 inet manual
up ifconfig eth0 up
```

```
auto eth1
iface eth1 inet manual
up ifconfig eth1 up
```

```
auto eth2
iface eth2 inet manual
up ifconfig eth0 up
```

```
up route add -net 10.1.200.0/24 dev eth0
up route add -net 172.16.2.0/24 dev eth1
up route add -net 192.168.2.0/24 dev eth2
```

```
gedit /etc/sysctl.conf
net.ipv4.ip_forward=1
net.ipv4.conf.eth0.proxy_arp=1
net.ipv4.conf.eth1.proxy_arp=1
net.ipv4.conf.eth2.proxy_arp=1
```

```
sudo apt-get install iptables-persistent
sudo /etc/init.d/iptables-persistent save
sudo /etc/init.d/iptables-persistent reload
```



## CHAPTER 4

### RESULTS

This chapter shows the snapshots taken for VBOX1 → VBOX3 Ping with highlighting TTL Value. It also shows wireshark report generation which is captured on VBOX4

#### 1. SELinux Disabled

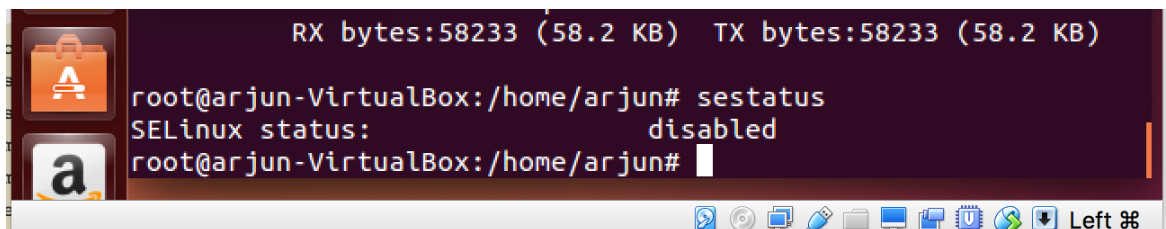


Figure 4.1: SELinux Disabled

#### 2 Ping from VBOX1 to VBOX3

Before and after changes in IPTABLE forwarding.

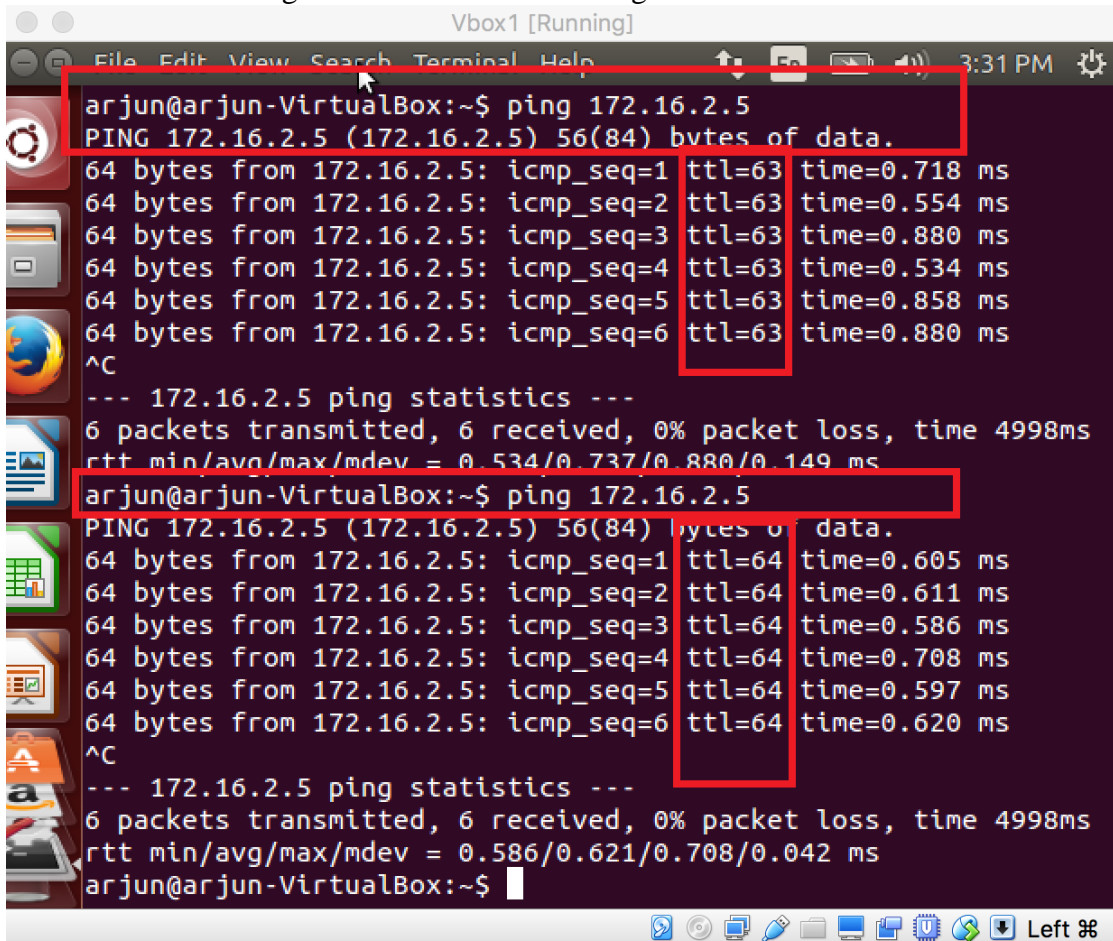
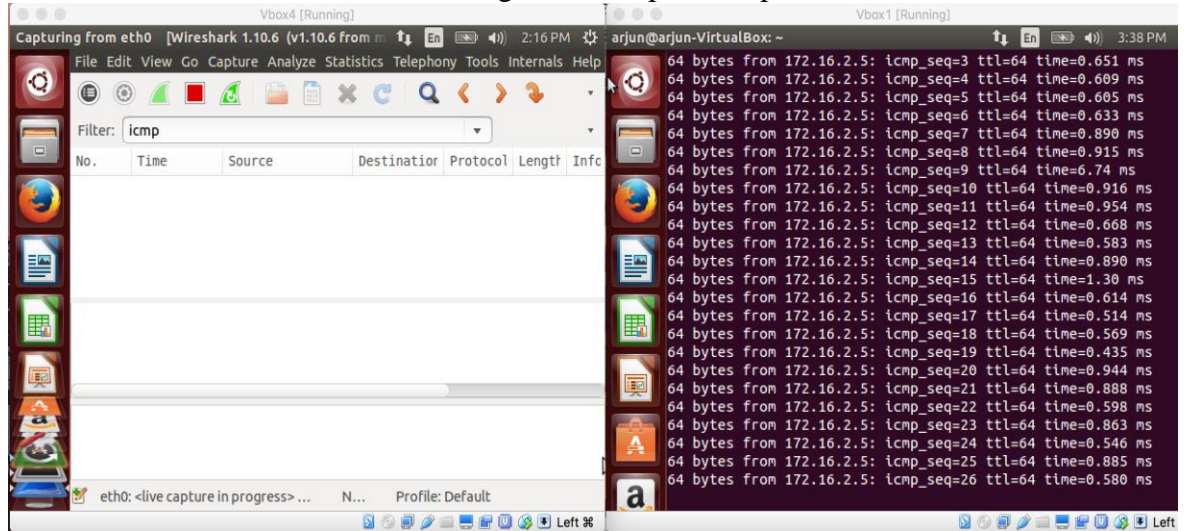


Figure 4.2: Ping from VBOX1 to VBOX3

### 3. Packet from VBOX1 to VBOX2 mirror on VBOX4 Management Interface

Before Executing Part5 Script of chapter 3



After Executing Part 5 Script of chapter 3

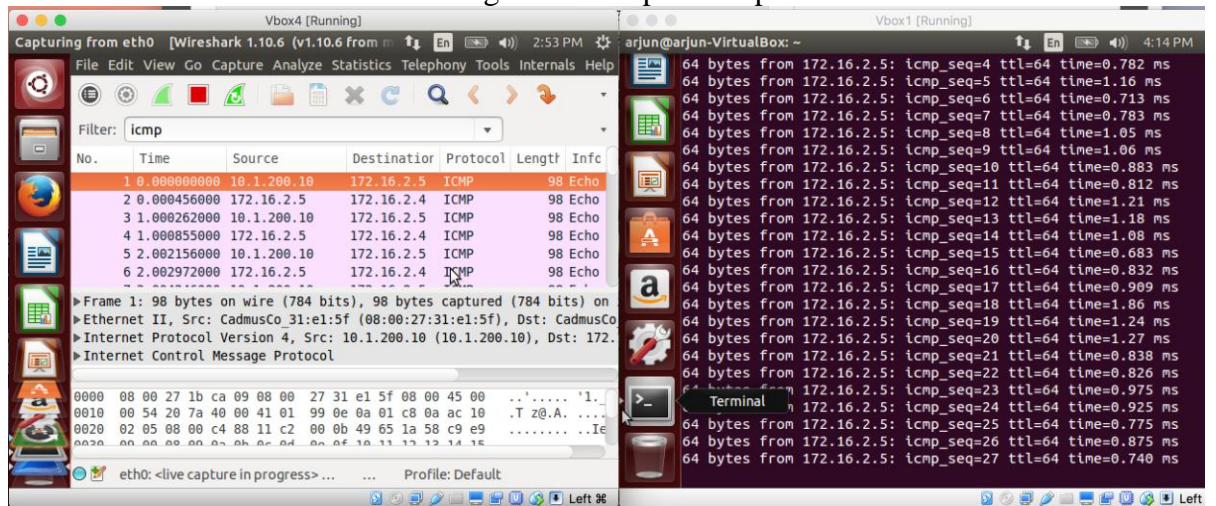


Figure 4.3: Mirror packet on VBOX4 Management Interface



#### 4. TCPDUMP and wireshark capture at VBOX3

- VBOX1 is Pinging to VBOX3.
- I run below command on VBOX3 to capture traffic via TCPDUMP  
tcpdump -i eth0 icmp
- From the capture it does not show any IP address of VBOX2. So VBOX2 behaves as a wire.

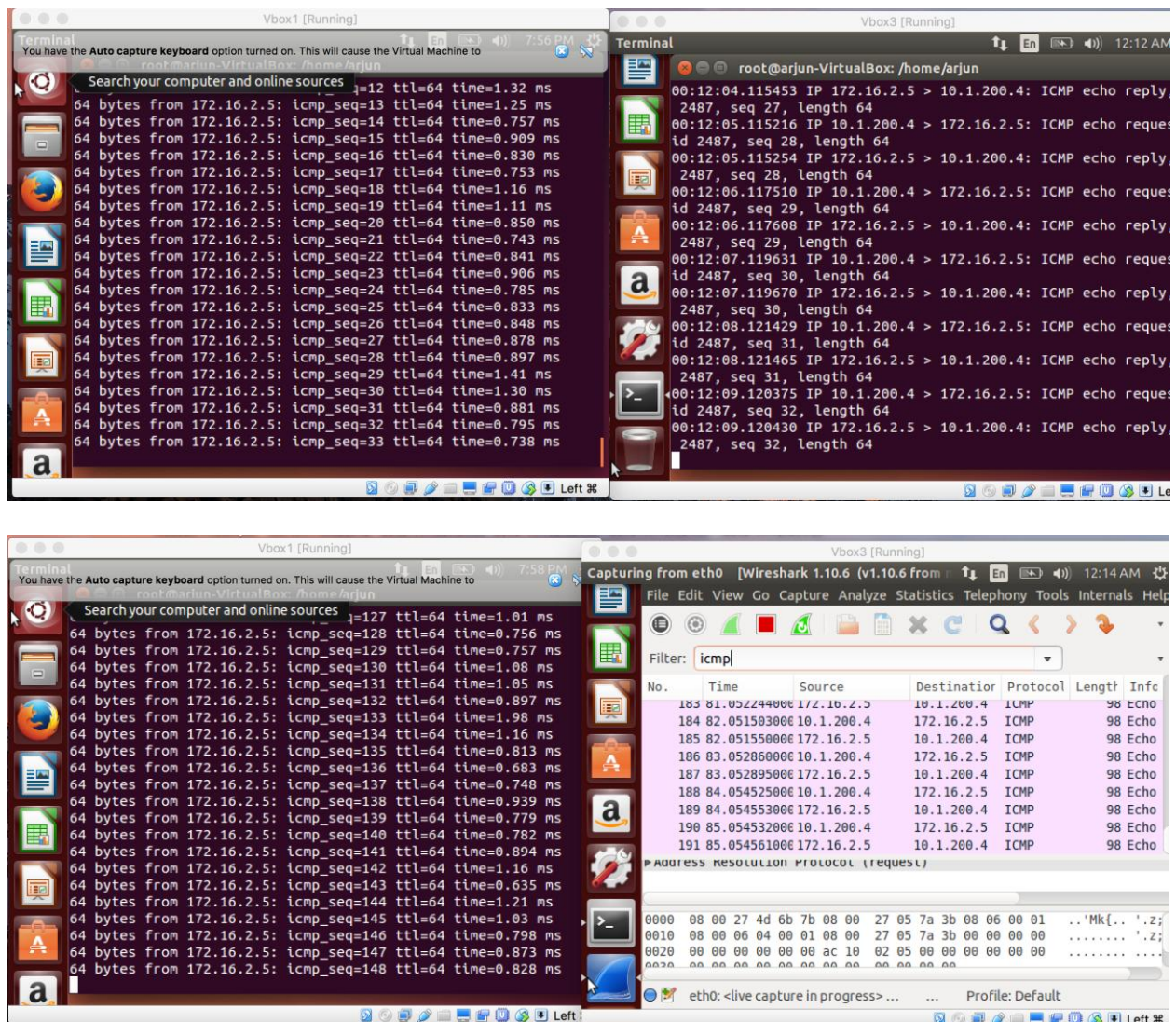


Figure 4.4: TCPDUMP and Wireshark capture at VBOX3



## 5. TCPDUMP and wireshark capture at VBOX4

- VBOX1 is Pinging to VBOX3.
- I run below command on VBOX4 to capture traffic via TCPDUMP  
tcpdump -i eth0 icmp
- VBOX4 is behaving as a Management Interface

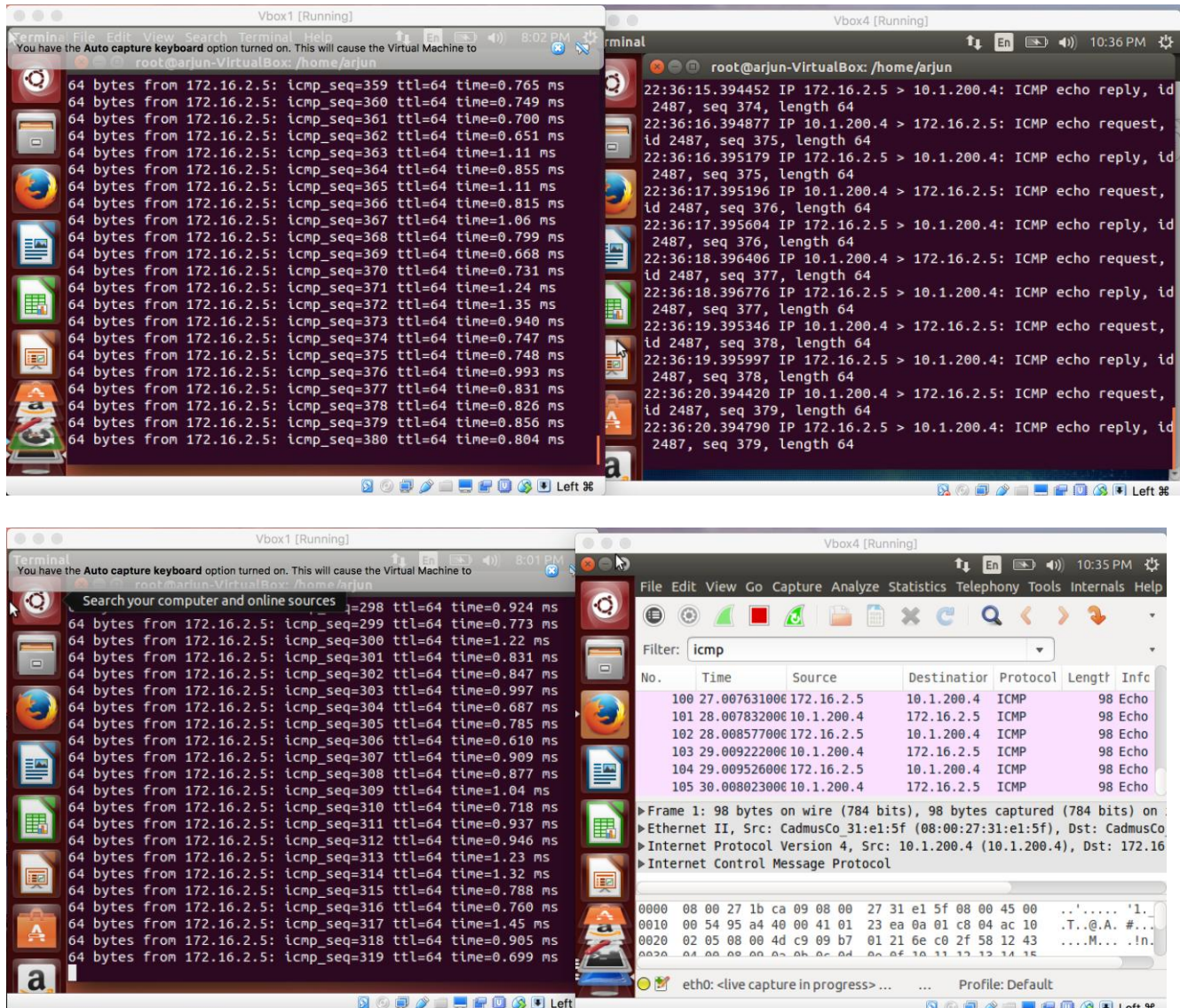


Figure 4.5: TCPDUMP and Wireshark capture at VBOX4

## 6. GEOIP Reports Generation

I uploaded GEOIP database on Wireshark and surf the webpage. It is showing location of city, country, latitude, and longitude with all listed public IP address.

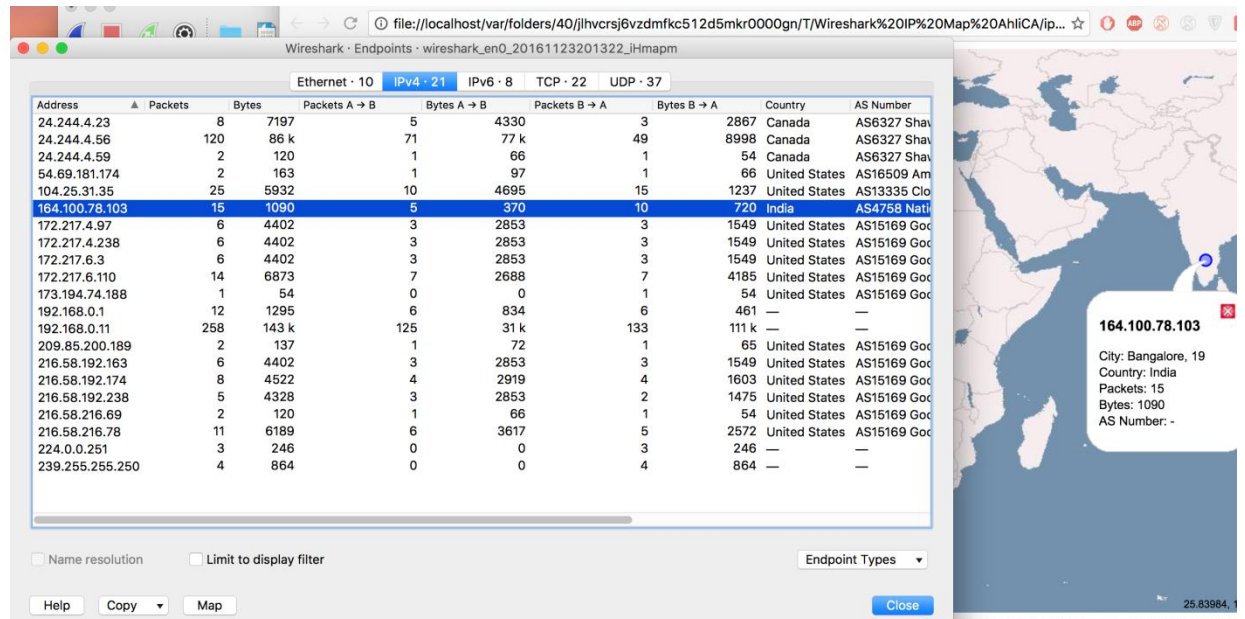


Figure 4.6: GEOIP Reports Generation

## 7. Basic Reports Generation

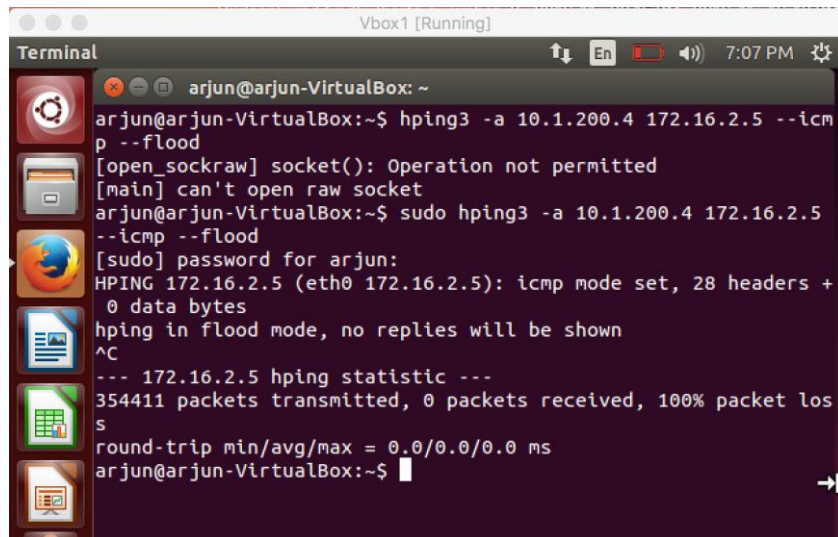
I used Hping3 for Generation of flood traffic from VBOX1 to VBOX3. Wireshark running within VBOX4 is capturing traffic.

```

Vbox1 [Running]
Terminal
root@arjun-VirtualBox: /home/arjun
10 packets transmitted, 10 received, 0% packet loss, time 9017m
s
rtt min/avg/max/mdev = 0.816/1.741/3.334/0.925 ms
root@arjun-VirtualBox: /home/arjun# hping3 -a 10.1.200.9 172.16.
2.5 --icmp --flood
HPING 172.16.2.5 (eth0 172.16.2.5): icmp mode set, 28 headers +
0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.16.2.5 hping statistic ---
3548502 packets transmitted, 0 packets received, 100% packet lo
ss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@arjun-VirtualBox: /home/arjun#

```

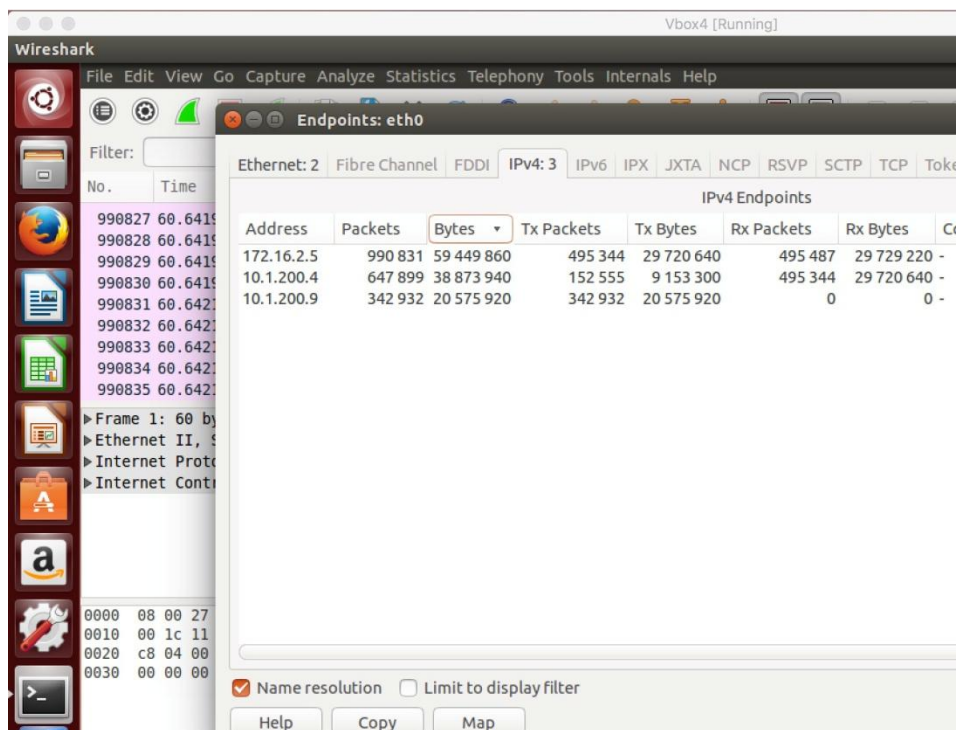




```
arjun@arjun-VirtualBox: ~  
arjun@arjun-VirtualBox:~$ hping3 -a 10.1.200.4 172.16.2.5 --icmp --flood  
[open_sockraw] socket(): Operation not permitted  
[main] can't open raw socket  
arjun@arjun-VirtualBox:~$ sudo hping3 -a 10.1.200.4 172.16.2.5 --icmp --flood  
[sudo] password for arjun:  
HPING 172.16.2.5 (eth0 172.16.2.5): icmp mode set, 28 headers + 0 data bytes  
hping in flood mode, no replies will be shown  
^C  
--- 172.16.2.5 hping statistic ---  
354411 packets transmitted, 0 packets received, 100% packet loss  
round-trip min/avg/max = 0.0/0.0/0.0 ms  
arjun@arjun-VirtualBox:~$
```

Figure 4.7.1: Hping3 for Generation of flood traffic

### Top Talkers (Listed by Size of Bytes)



IPv4 Endpoints							
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	
172.16.2.5	990 831	59 449 860	495 344	29 720 640	495 487	29 729 220	-
10.1.200.4	647 899	38 873 940	152 555	9 153 300	495 344	29 720 640	-
10.1.200.9	342 932	20 575 920	342 932	20 575 920	0	0	-

Figure 4.7.2: Top Talkers

### Packet counts by Protocols (As only Ping is used, it shows ICMP traffic)

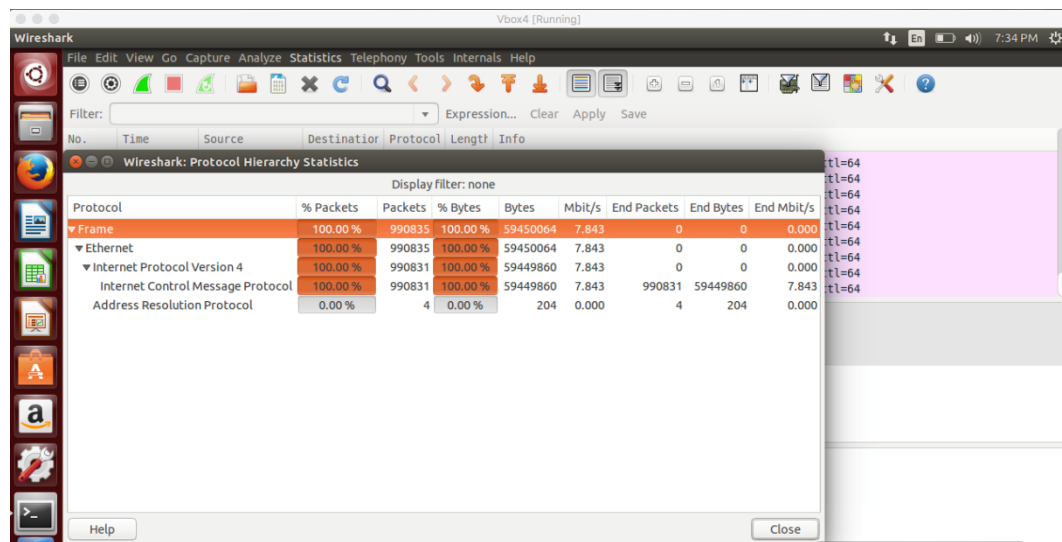


Figure 4.7.3: Packet Counts by Protocols

### Packet Length Statistics

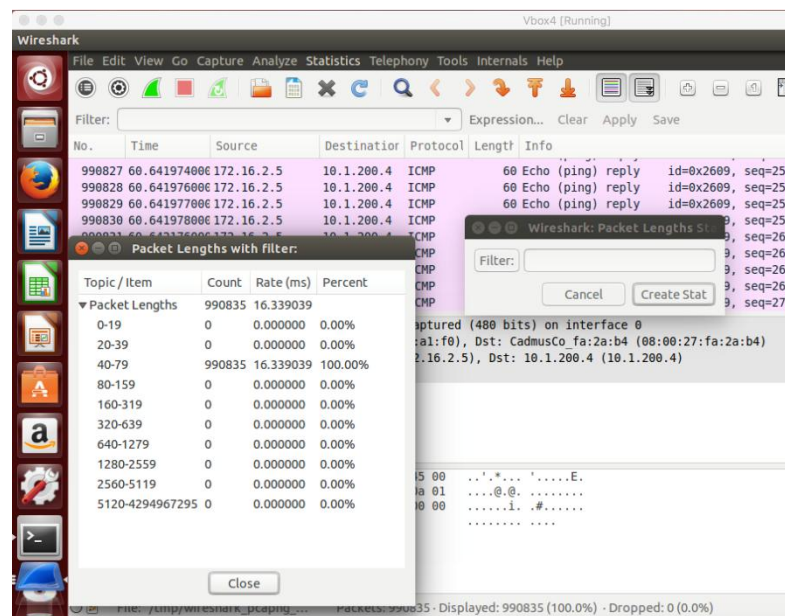


Figure 4.7.4: Packet Length Statistics

## Packet Length Statistics by Protocol

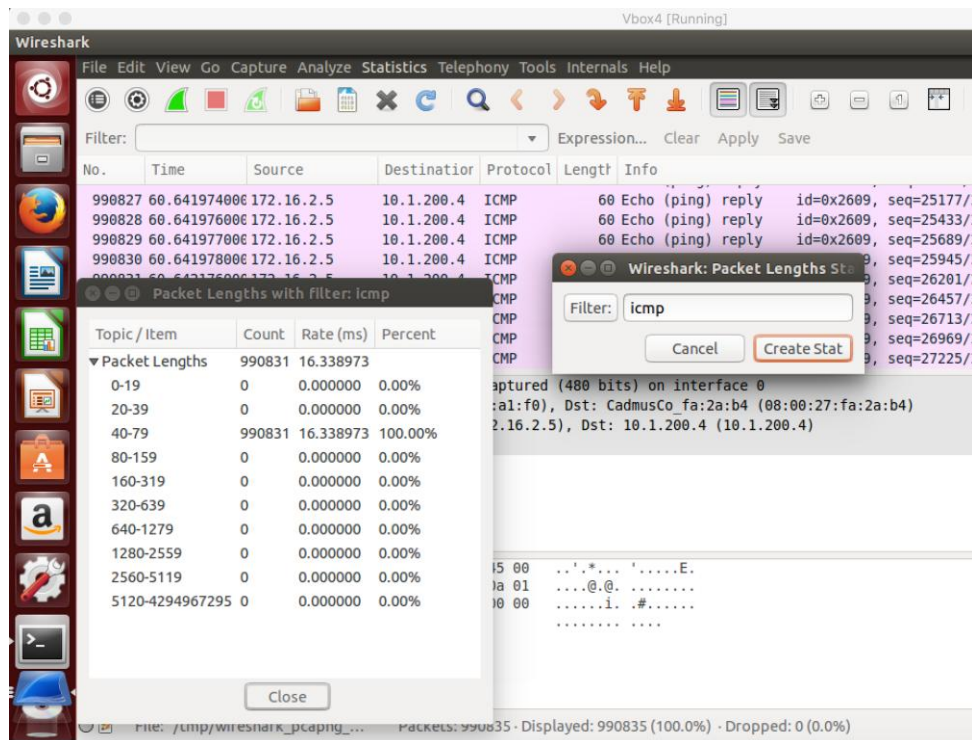


Figure 4.7.5: Packet Length Statistics by Protocol



## **CHAPTER 5**

---

### **CONCLUSION**

I modified Linux operating system to capture traffic passing through two of its interfaces. As per the results packet is just travelling on the wire without crossing any interfaces because hop count is not changing. Intermediate VBOX2 is not detectable because of these changes.

Packets are also transferred on management interface, where the IPS (wireshark/snort) solution will be built as the next stage of Project for future scope.

## REFERENCES

---

### Website Link:

1. <https://opensource.com/resources/what-is-linux>
2. [https://en.wikipedia.org/wiki/Ubuntu\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Ubuntu_(operating_system))
3. <https://www.virtualbox.org/manual/ch06.html>
4. <https://www.virtualbox.org/manual/ch01.html#idm26>
5. [https://en.wikipedia.org/wiki/Intrusion\\_detection\\_system](https://en.wikipedia.org/wiki/Intrusion_detection_system)
6. [https://en.wikipedia.org/wiki/Snort\\_\(software\)](https://en.wikipedia.org/wiki/Snort_(software))
7. <https://en.wikipedia.org/wiki/Wireshark>
8. <https://www.digitalocean.com/community/tutorials/how-the-iptables-firewall-works>
9. [http://www.linuxtopia.org/Linux\\_Firewall\\_iptables/x4799.html](http://www.linuxtopia.org/Linux_Firewall_iptables/x4799.html)
10. <https://www.tummy.com/articles/networking-basics-how-arp-works/>
11. <https://help.ubuntu.com/community/IptablesHowTo>
12. <https://help.ubuntu.com/community/VirtualBox/Networking>
13. <http://blogging.dragon.org.uk/setup-a-test-networking-lab-with-virtualbox/>
14. <http://myogyisg.blogspot.ca/2012/01/linux-iptables-routing-snat-masquerade.html>
15. <http://tlug.ss.narkive.com/mrfX145J/setting-up-virtual-box-to-snat-dnat>
16. <https://forums.virtualbox.org/viewtopic.php?f=7&t=37133>
17. <http://serverfault.com/questions/284056/simulated-nat-traversal-on-virtual-box>
18. <http://servertopic.com/topic/A9Rp-simulated-nat-traversal-on-virtual-box>
19. <http://unix.stackexchange.com/questions/74663/virtualbox-nat-host-only-adapter>
20. <https://www.tolaris.com/2009/03/05/using-host-networking-and-nat-with-virtualbox/>
21. <http://linux-training.be/networking/ch13.html>
22. <https://gtalug.org/pipermail/legacy/2010-January/048633.html>
23. <https://community.oracle.com/docs/DOC-890092>
24. <http://www.howopensource.com/2011/06/how-to-use-virtualbox-in-terminal-commandline/>

## References

---

25. <http://linuxpitstop.com/install-and-use-command-line-tool-vboxmanage-on-ubuntu-16-04/>
26. <https://www.unixmen.com/install-ubuntu-14-04-3-lts-virtualbox/>
27. <https://technology.amis.nl/2014/01/27/a-short-guide-to-networking-in-virtual-box-with-oracle-linux-inside/>
28. <https://openmaniak.com/tcpdump.php>
29. <http://networkengineering.stackexchange.com/questions/10530/ping-between-different-subnet-across-a-link>
30. [http://archive.openflow.org/wk/index.php/OpenFlow\\_Tutorial](http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial)
31. <https://blog.wireshark.org/2010/02/running-wireshark-as-you/>

### **Books:**

32. Charles M. Kozierok, The TCP/IP Guide, Version 3.0, 2005
33. Laura chappell, Wireshark 101 Essential Skills for Network Analysis, 2013.

### **Video Link:**

34. <https://www.youtube.com/watch?v=WXhpSRaae7w>
35. <https://www.youtube.com/watch?v=1PsTYAd6MiI>
36. <https://www.youtube.com/watch?v=ZD0-LOCKpO4>
37. <https://www.youtube.com/watch?v=XKfhOQWrUVw>
38. <https://www.youtube.com/watch?v=ZtCWXdIKE0>
39. <https://www.youtube.com/watch?v=Vf2S-XeZoeQ>
40. <https://www.youtube.com/watch?v=D08catMKcRg>
41. [https://www.youtube.com/results?search\\_query=wireshark+IO+graphs](https://www.youtube.com/results?search_query=wireshark+IO+graphs)
42. <https://www.youtube.com/watch?v=hrExIIJYJho>