Praarthana Ramakrishnan (praartr)

Advanced Network Security

Project 2

TASK 1: Writing Packet Sniffing Program

1. **Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial.**

Solution:

1. Set the device : To choose which interface to sniff on

2 ways to do it:

* User defines the device using string
* pcap gives name of interface using

dev = pcap\_lookupdev(errbuf); If this fails, error message is saved in errbuf.

1. Open the device for sniffing

* pcap\_t \*pcap\_open\_live( char \*device, int snaplen, int promisc, int to\_ms, char \*ebuf)

where

1. char \* device - device that we want to stiff on.
2. int snaplen - maximum number of bytes captured by pcap
3. int promisc – 1 for promiscuous mode and 0 for non-promiscuous mode
4. int to\_ms – read time out in milliseconds
5. char \*ebuf – saves error message
6. Filter Traffic

* Compile the filter using

Int pcap\_compile(pcap\_t \*p, struct bpf\_program \*fp, char \*str, int optimize, bpf\_u\_int32 netmask)

Where

1. pcap\_t \*p - session handle
2. struct bpf\_program \*fp - to strore the compiler version of the filter
3. char \* str - the expression
4. int optimize – 1 means network should be optimized

* pcap\_lookupnet() - returns one of IPV4 network numbers and mask.
* int pcap\_setfilter(pcap\_t \*p, struct bpf\_program \*fp)

where

1. pcap\_t \*p – session handler
2. struct bpf)program \*fp – reference to compilerd version of expression
3. Sniffing

* Capture a single packet at a time using:

u\_char \*pcap\_next(pcap\_t \*p, struct pcap\_pkthdr \*h);

where

1. pcap\_t \*p – session handler
2. struct pcap\_pkthdr \*h – pointer to general information about the packet

* Perform a loop that waits for n number of packets to be sniffed using :

Int pcap\_loop(pcap\_t \*p, int cnt, pcap\_handler callback, u\_char \*user)

Where

1. pcap\_t \*p – session handler
2. int cnt – how many packets to sniff
3. pcap\_handler callback – the name of callback function
4. Close the sniffer session

* pcap\_close()

1. **Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?**

Solution:

STEP 1: First, we compile sniffex.c like shown in Figure 1 below.

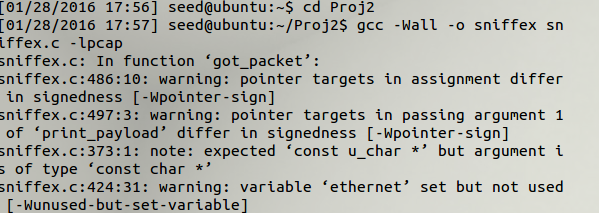


Figure 1

Once it is compiled, we try to run sniffex without root privilege as shown as Figure 2

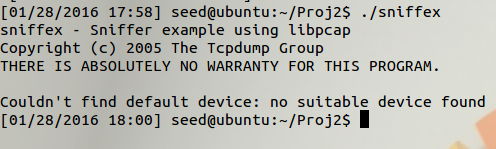


Figure 2

The message is printed “Couldn’t find default device: no suitable device found “because there is no root privilege. The error message is saved in the errbuf and the error message is printed.

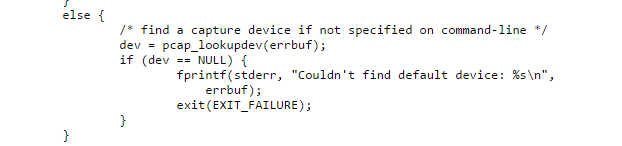


Figure 3

Successfully runs sniffex with root privilege as shown in Figure 4 below.

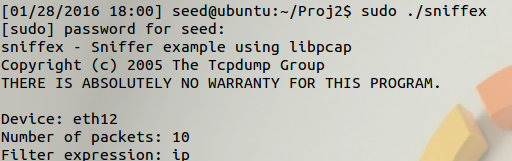


Figure 4

1. **Please turn on and turn off the promiscuous mode in the sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you demonstrate this.**

Solution:

Promiscuous mode and Non-Promiscuous mode:

In non-promiscuous sniffing, a host is sniffing only traffic that is directly related to it. Only traffic to, from, or routed through the host will be picked up by the sniffer.

Promiscuous mode, sniffs all traffic on the wire.

I set up two virtual machines and all in running mode. Then I turned on the promiscuous mode.

Turn on virtual machine’s network setting as host-only to be under the same network.

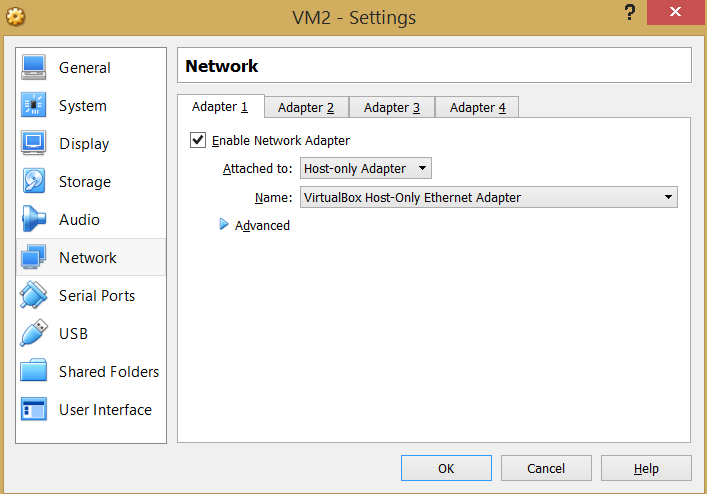


Figure 5

Type “ifconfig” to find the IP addresses of the two virtual machines.

Now we have two virtual machines with IP addresses as: Ubuntu (192.168.56.101) and Ubuntu Clone (192.68.56.102).

The communication is initiated using ping command as shown as figure 6.

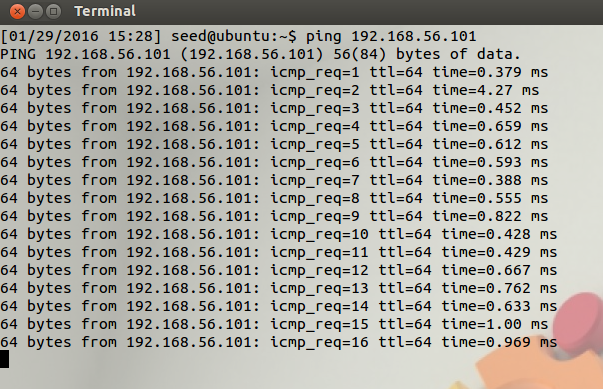


Figure 6

Now, the communication is good. The filter expression is changed to “icmp” since ping uses “icmp”.

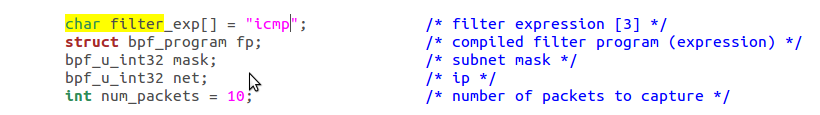


Figure 7

Now the promiscuous mode is ON as shown in figure 8 and figure 9 and run sniffex from Ubuntu with root privilege and capture all traffic from the network.

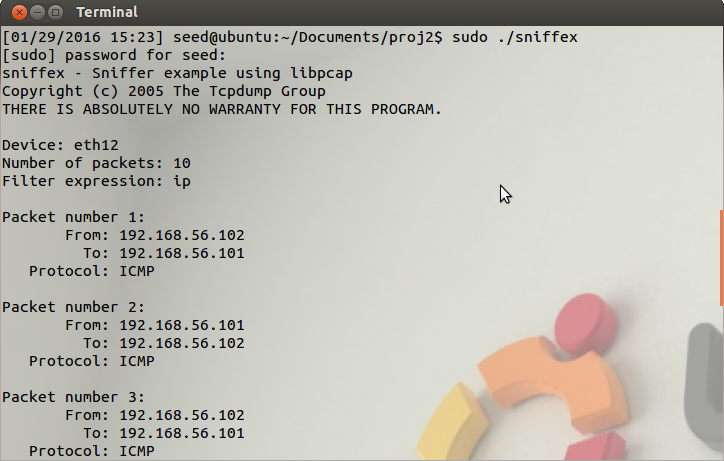


Figure 8

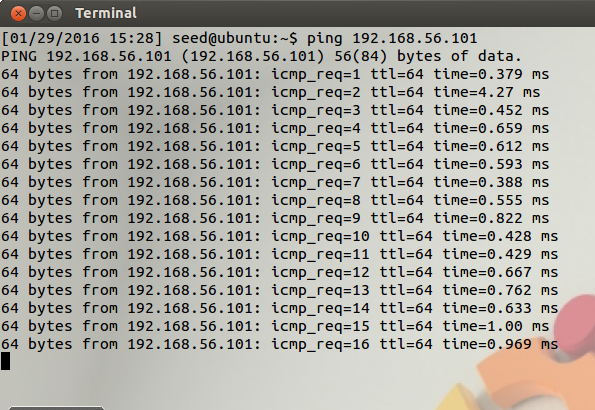


Figure 9

Now , the promiscuous mode is turned OFF by changing the argument to 0 as shown below in Figure 10.

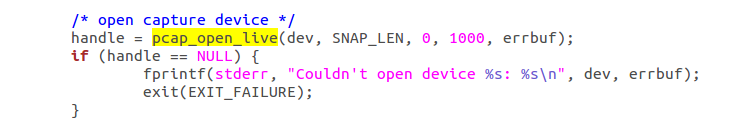


Figure 10

Since the promiscuous mode is turned OFF, it can only capture packets from or to it.

1. **Please write filter expressions to capture each of the followings. In your lab reports, you need to include screen dumps to show the results of applying each of these filters.**

* **Capture the ICMP packets between two specific hosts.**
* **Capture the TCP packets that have a destination port range from to port 10 – 100.**

ICMP packets are already captured before as shown in Figure 7 and Figure 11.

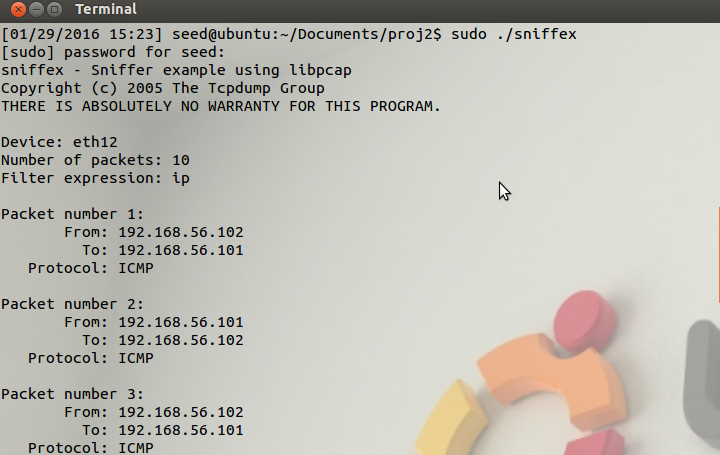


Figure 11

To capture TCP packets that have a destination port range from 10-100, we need to use filter expression as shown in Figure 12.

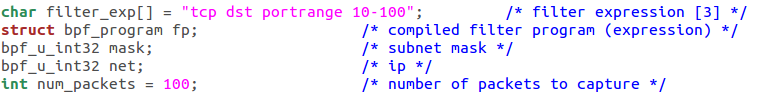


Figure 12

The TCP packets are successfully being captured as shown in Figure 13.

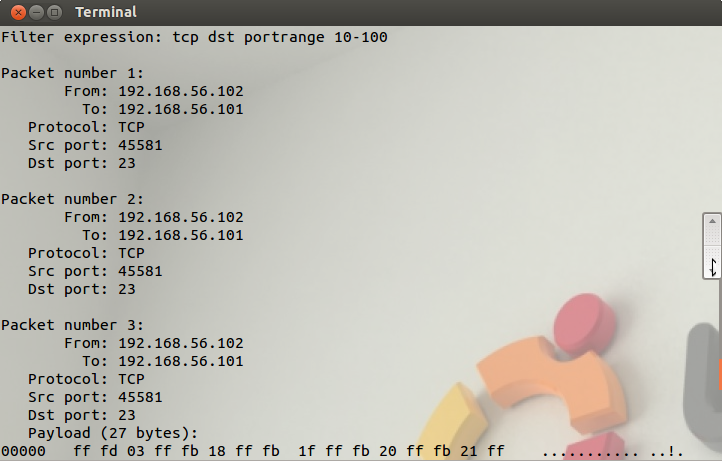


Figure 13

**5) Please show how you can use sniffex to capture the password when somebody is using telnet on the network that you are monitoring. You may need to modify the sniffex.c a little bit if needed. You also need to start the telnetd server on your VM. If you are using our pre-built VM, the telnetd server is already installed; just type the following command to start it.**

**% sudo service openbsd-inetd start**

The filter expression is changed to “tcp and port 23” to capture telnet packets as shown in Figure 14.

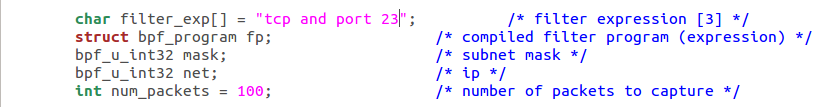


Figure 14

Then we start telnet server on VM as shown in Figure 15. This is how we use sniffex to capture password when somebody is using telnet on the network.

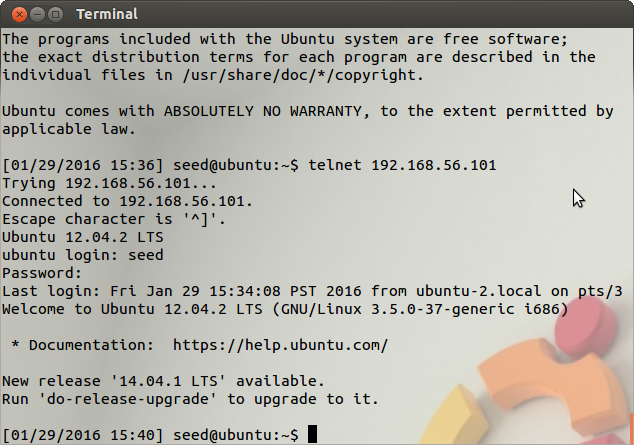


Figure 15