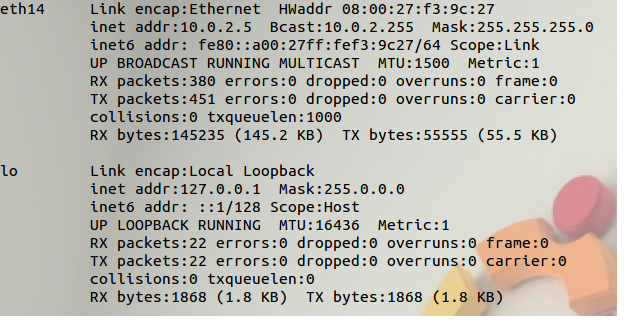
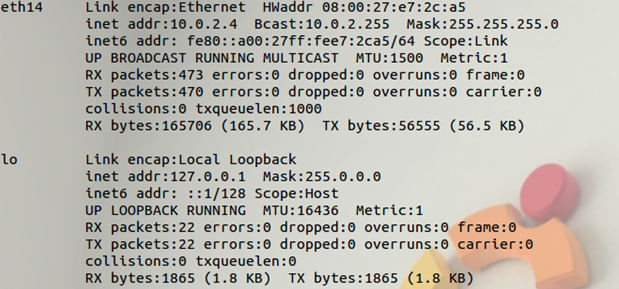
PHONG TRUONG

1.Verifying the network:

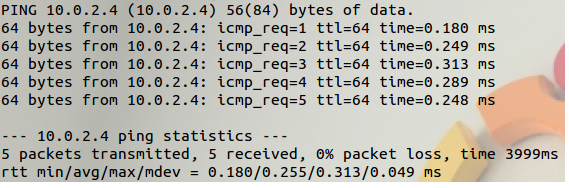
VM1

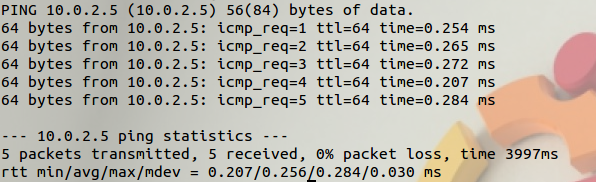


Clone:



Ping





2.

Setting the device:

First, we have the set the program to sniff on a device of our choosing by either passing the name of the device into the program or the program or the program will just set the device to sniff with a predetermined choice on its own.

Opening the device for sniffing:

We then create a sniffing session by calling the “pcap\_open\_live()” method which contains the following arguments: first, the name of the device; second, the maximum number of bytes to be captured; third, setting promiscuous mode true or false; fourth, the read time out in milliseconds; fifth, the string to store any error messages. Promiscuous mode sniffs all traffic on the connection while non-promiscuous mode only sniffs traffic related to the host. We also need to provide the type of link-layer header as not all devices have the same type. The program will fails if it doesn’t support the link-layer header type provided by the device.

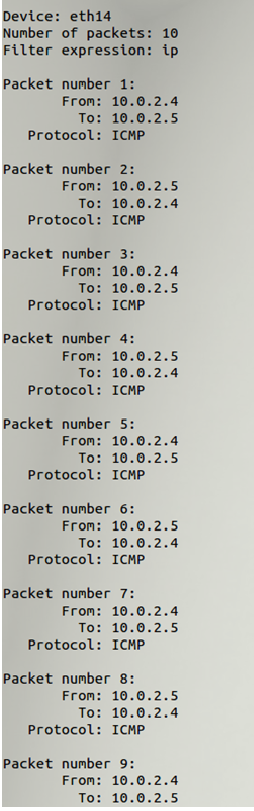
Filtering traffic:

After calling “pcap\_open\_live()”, we can apply a filter using pcap’s filter as it uses the BPF filter and much more efficient. Before applying the filter, we must compile first by calling “pcap\_compile()” which takes in arguments as the following; first, session handle; second, a reference to the place to store the compiled filter; third, the expression itself; fourth, whether or not to optimize the expression, and fifth, the network mask (ip address) the filter applies to. After, we can set the filter using “pcap\_setfilter(),” taking in arguments: first, the session handler, and second, a reference to the compiled filter.

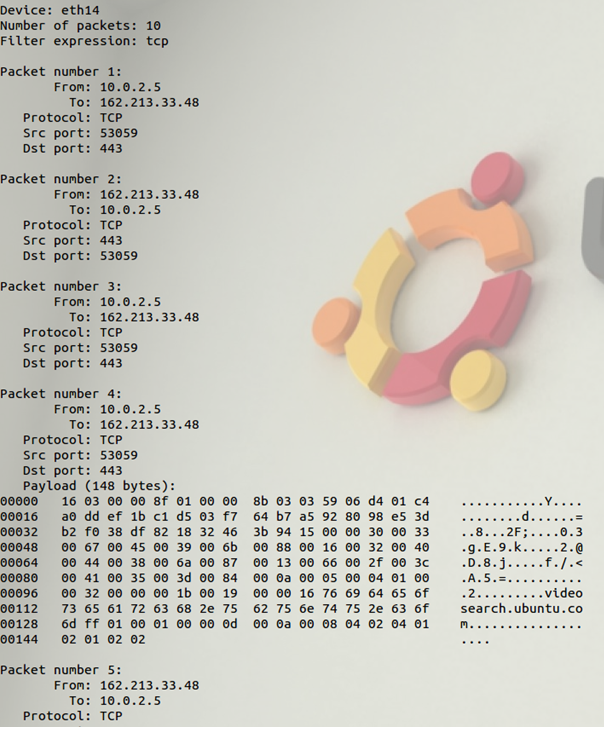
The actual sniffing:

To capture packets, we can use one of two methods: capturing a single packet at a time or looping to capture “n” number of packets at a time. First, “pcap\_next()” takes in two arguments (a session handler and pointer to a struct) and returns a char pointer to the packet. Second, we can use “pcap\_loop(),” taking in arguments (a session handler, number of packets, a callback function, and an user argument used in pcap\_loop and/or callback). Before looping, we must instantiate the callback function using “got\_packet(),” which takes in the following arguments (user defined args, a pointer to a struct to store, and a pointer to a char for the packet).

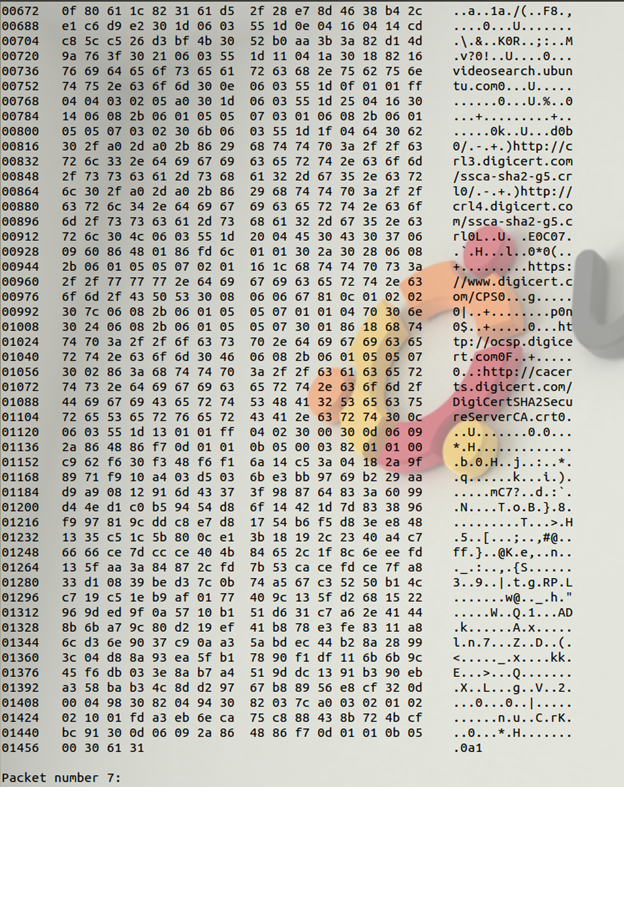
Running



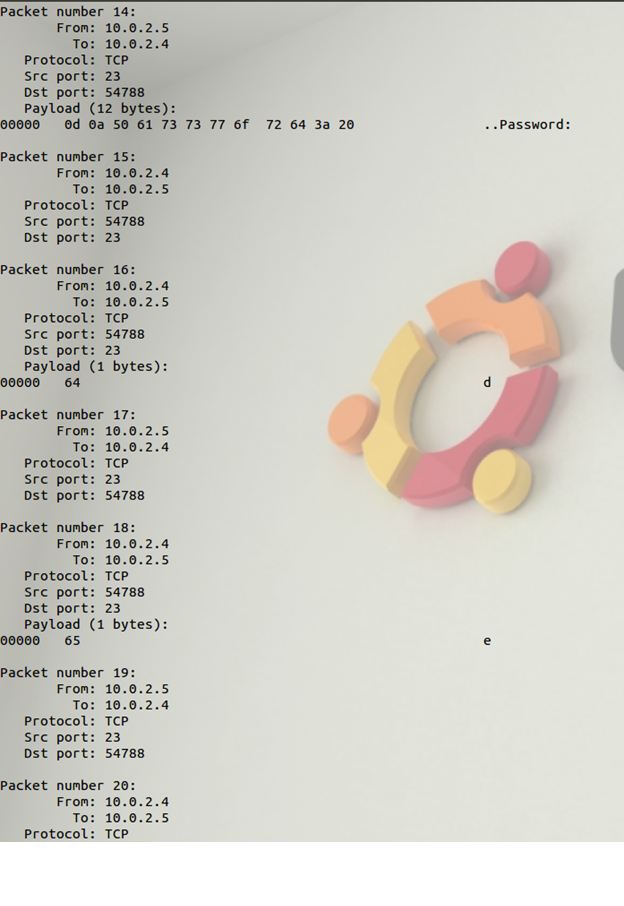
TCP Bytes of information are now shown

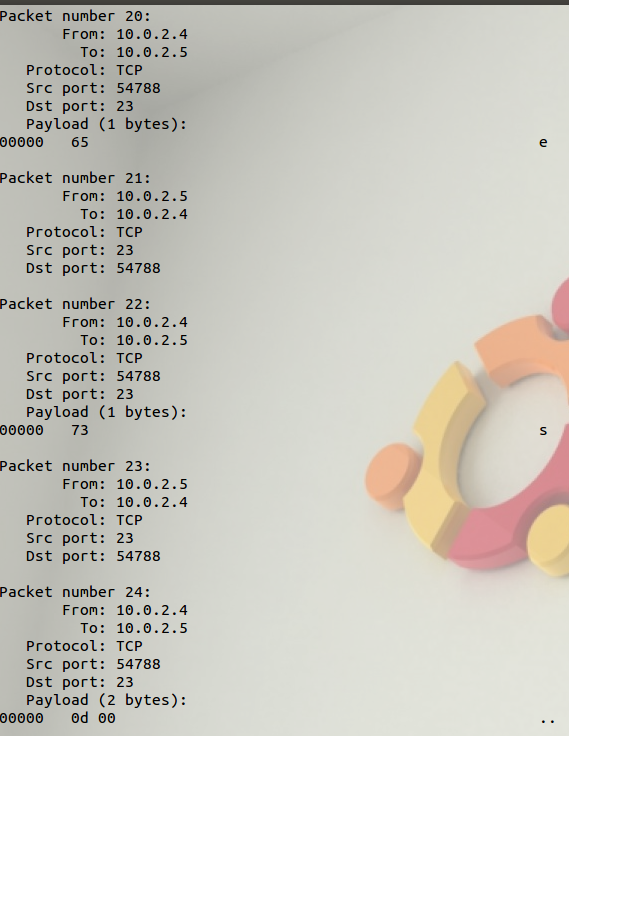






3. 3. Password Sniffing







Telnet does not seem to be a secure method of remotely accessing a system as somebody on the same network can sniff the packets to see the password.

4. SSH

No, we cannot find the user’s password as it is now encrypted using SSH so it appears as a messy jumble. SSH is therefore more secure than Telenet.

