Aryaman Mishra

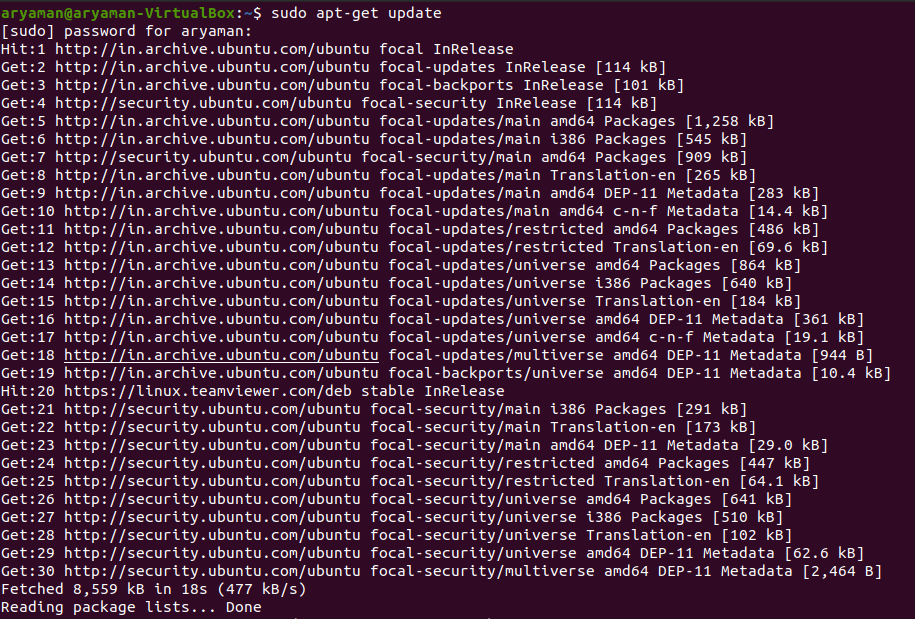
19BCE1027

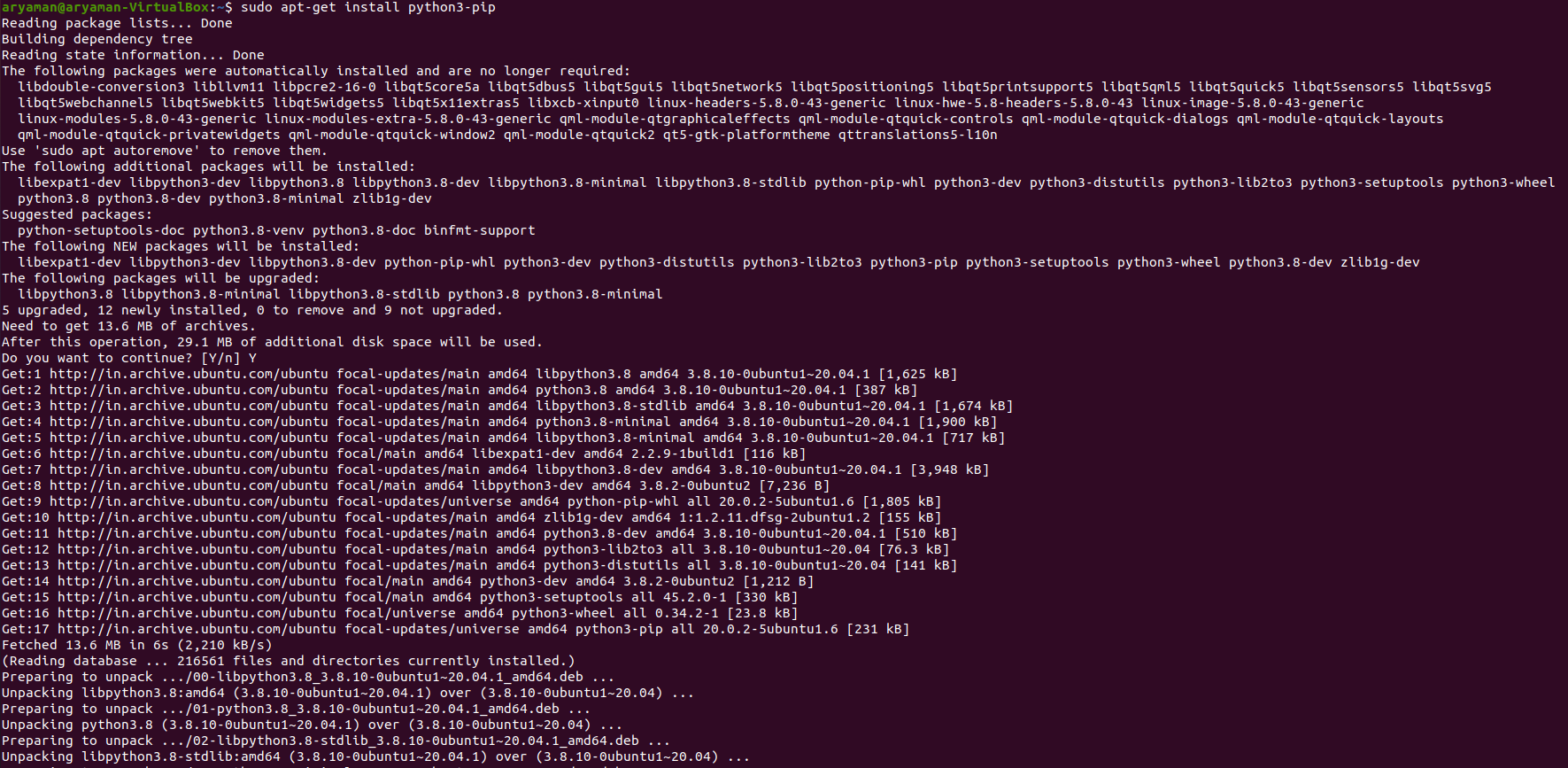
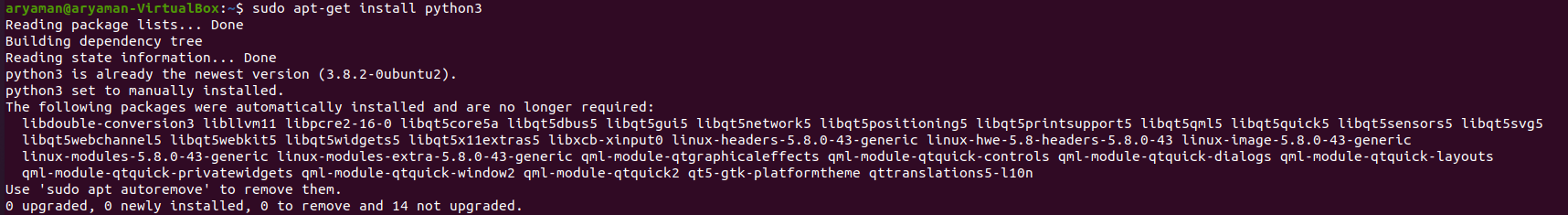
**SCAPY**

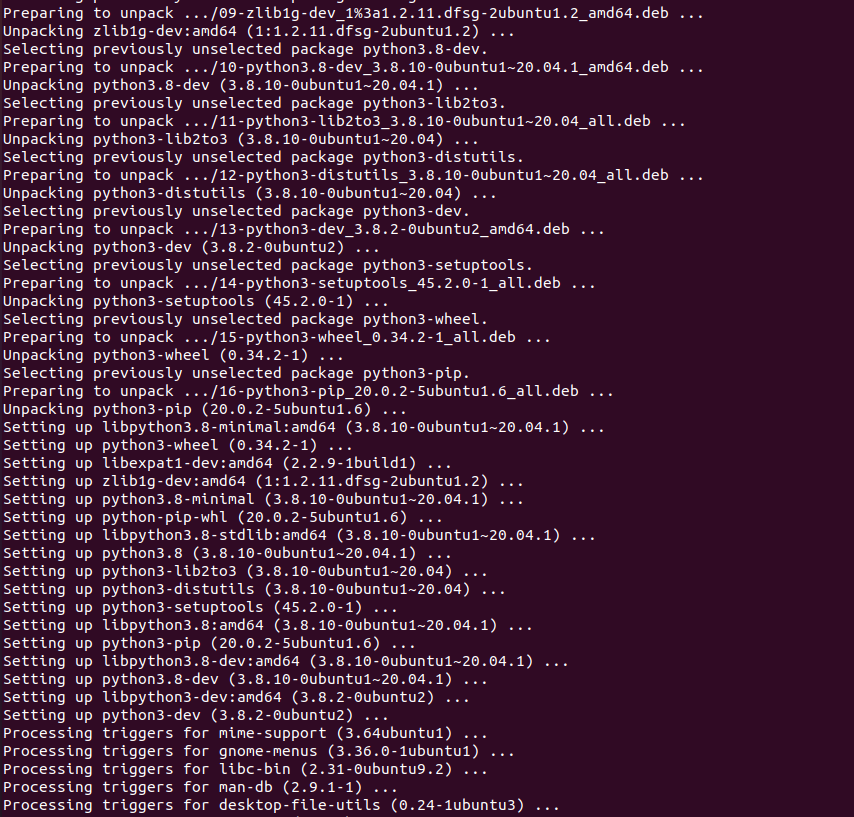
**TASK 1: Identifying the http site and provide URL**

<http://www.testingmcafeesites.com/>

The following steps describe how to install (or update) Scapy itself. Dependent on your platform, some additional libraries might have to be installed to make it actually work.

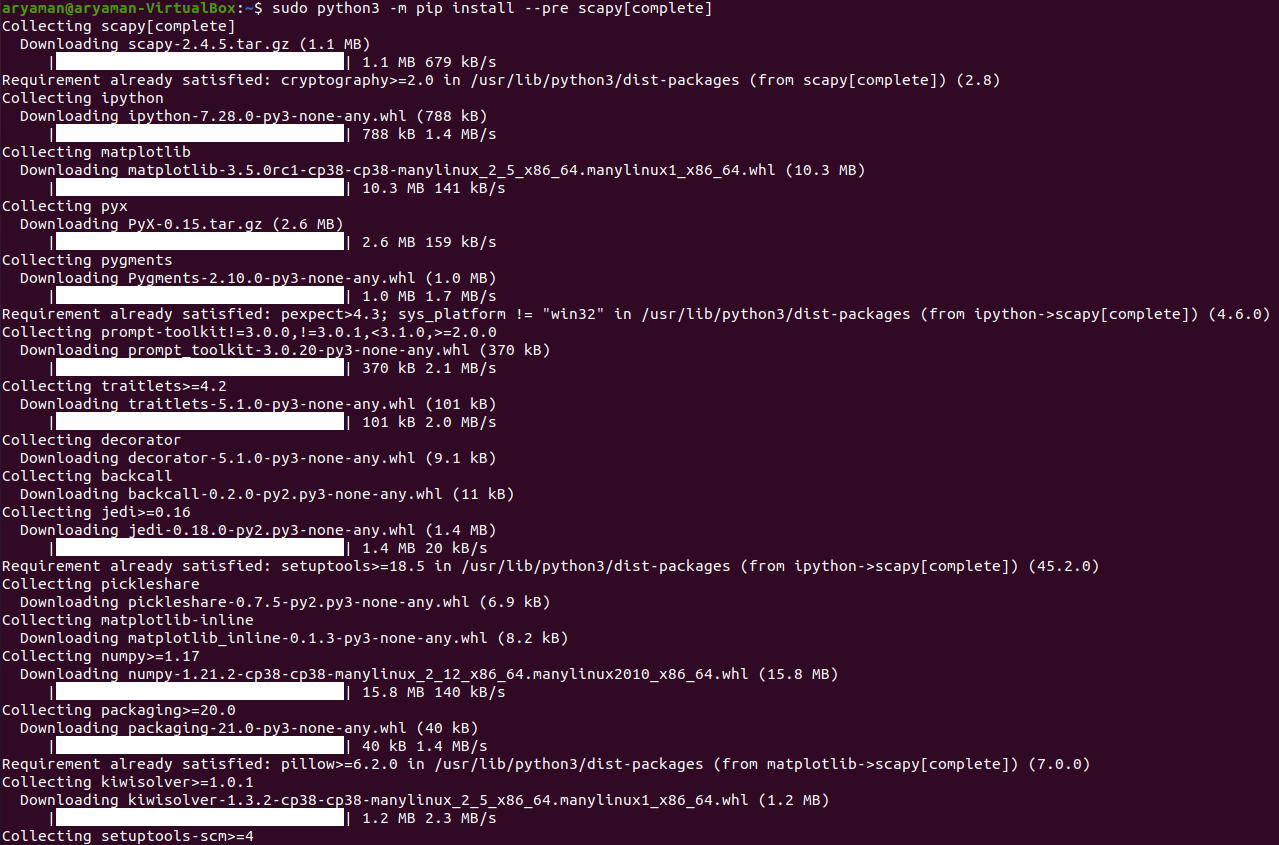
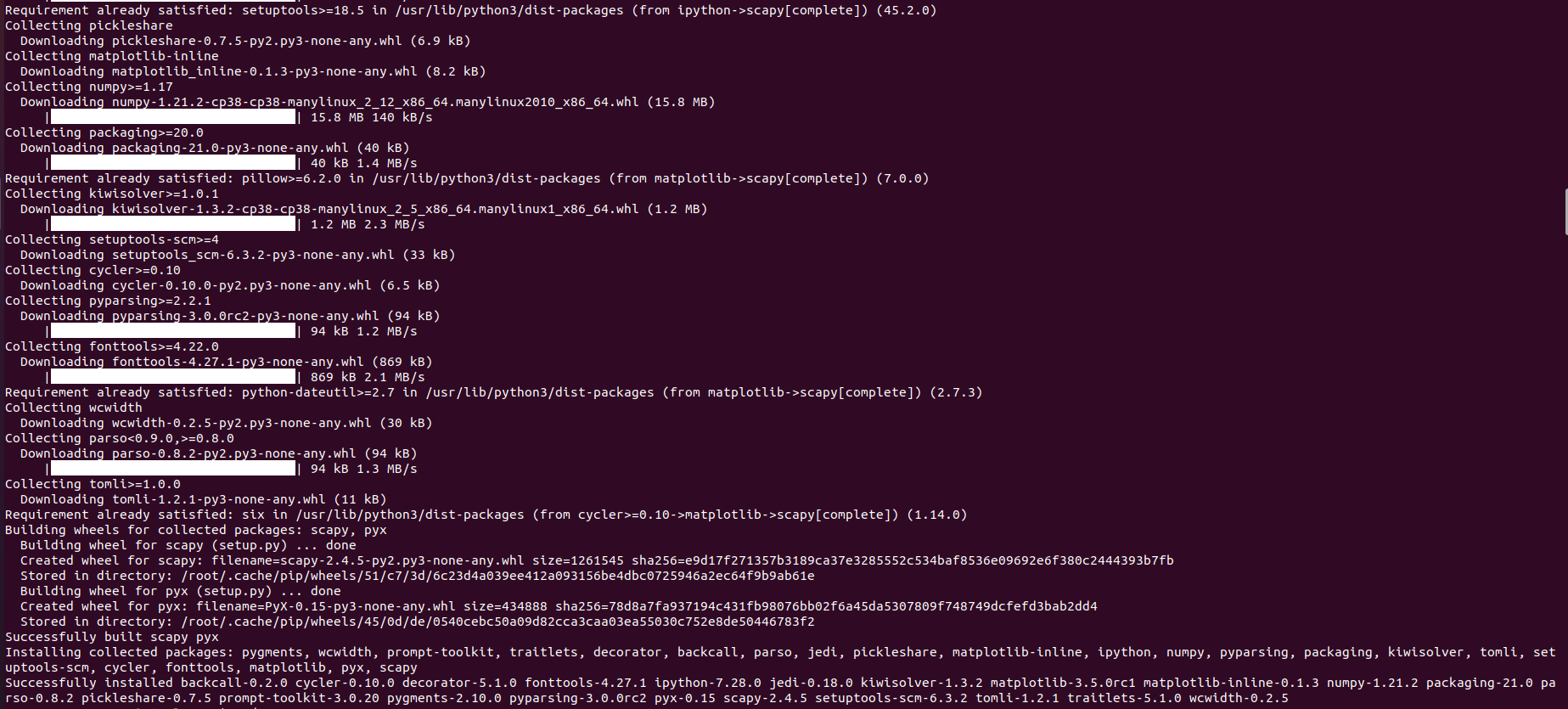


Make sure you have Python installed before you go on. 

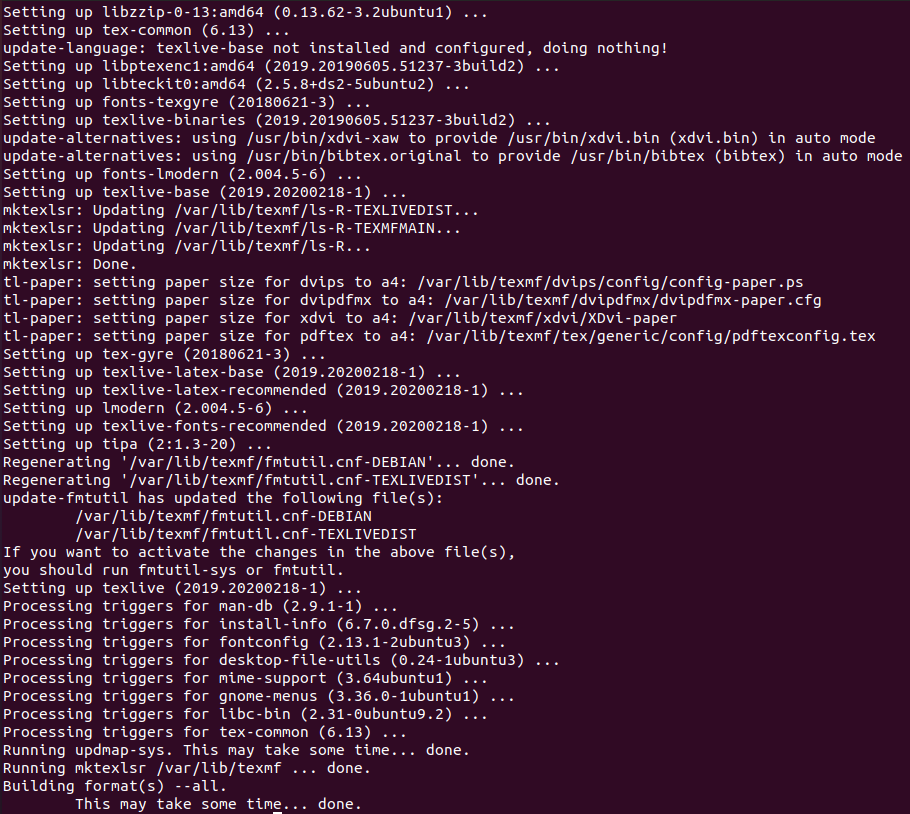
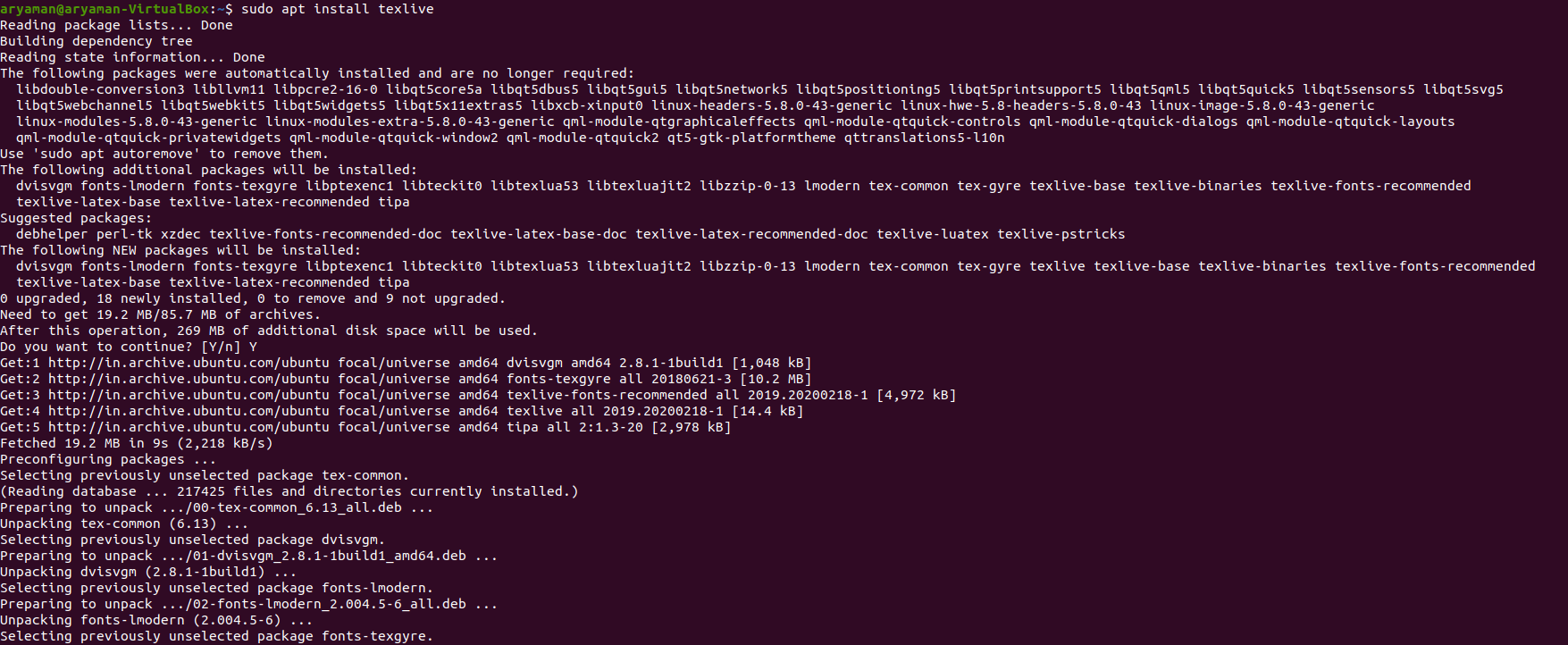


In fact, since 2.4.3, Scapy comes in 3 bundles:

| **Bundle** | **Contains** | **Pip command** |
| --- | --- | --- |
| Default | Only Scapy | pip install scapy |
| Basic | Scapy & IPython. **Highly recommended** | pip install --pre scapy[basic] |
| Complete | Scapy & all its main dependencies | pip install --pre scapy[complete] |



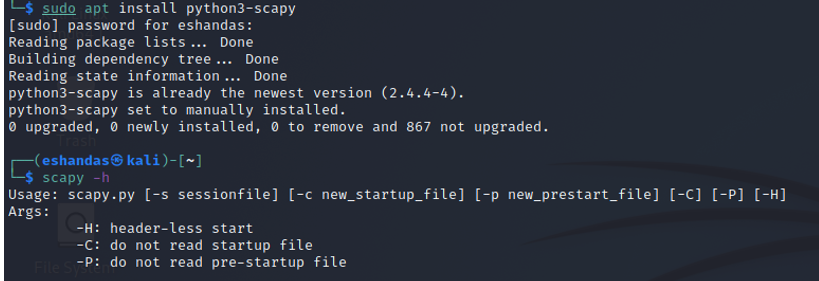
We can also install TexLive as optional dependancies to avoid errors.



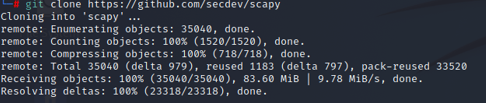
## Starting Scapy[ℑ](https://scapy.readthedocs.io/en/latest/usage.html" \l "starting-scapy" \o "Permalink to this headline)

Scapy’s interactive shell is run in a terminal session. Root privileges are needed to send the packets, so we’re using sudo here:

**INSTALLING SCAPY (METHOD 1)**



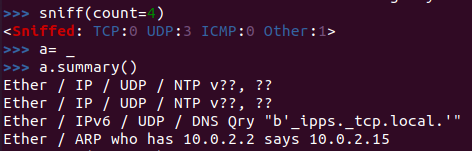
**INSTALLING SCAPY (METHOD 2)**



**RUN THE SCAPY TOOL**



We will start with sniffing 4 packets and using summary() function to view their information.Basically it shows the layer of the packets.We can see Ethernet frame on the network access layer,it’s an IP protocol on the Internet layer,at the transport layer,it’s UDP and we can view the Domain name(or DNS Query) on the Application layer.



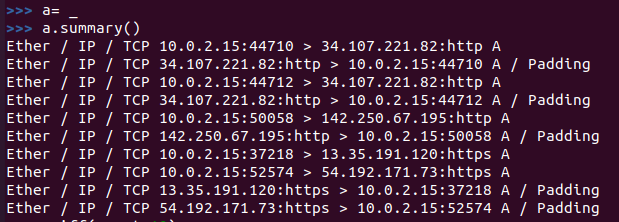
Scapy has sniffed 4 UDP packets in a single line.



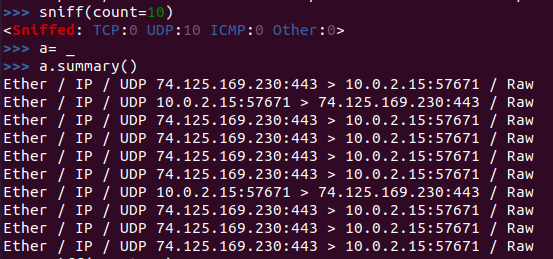
Scapy has sniffed 4 TCP packets in a single line.



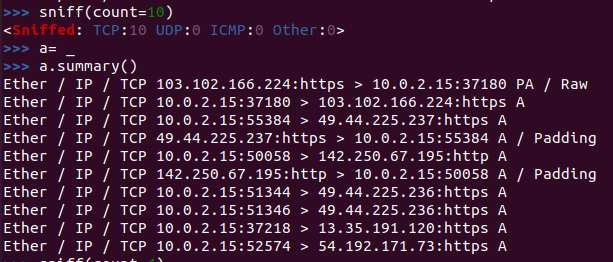
We can view the summary of those 10 TCP packets by storing them in a variable a and using the summary() function.



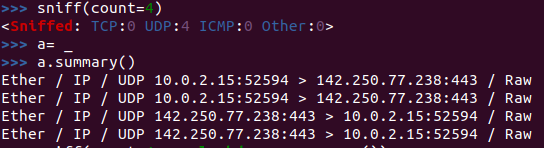
We can view the summary of those 10 UDP packets by storing them in a variable a and using the summary() function.



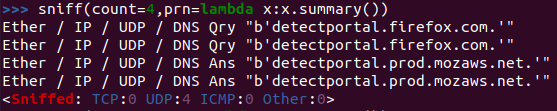
We can view the summary of those 10 TCP packets by storing them in a variable a and using the summary() function.

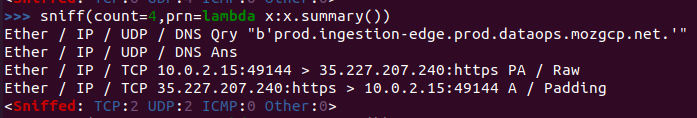


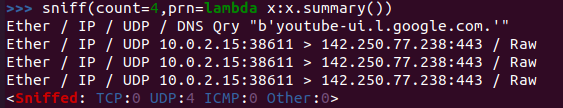
We can view the summary of those 4 UDP packets by storing them in a variable a and using the summary() function.



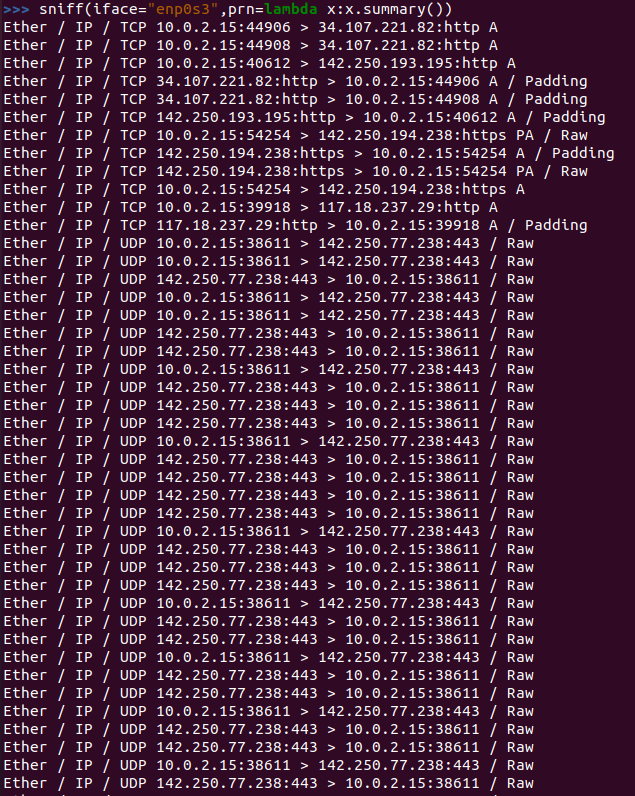
If we want the summary in a single command,we can use the lambda function to find the summary of n number of packets.



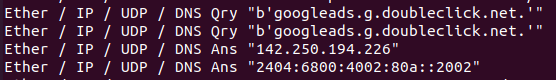


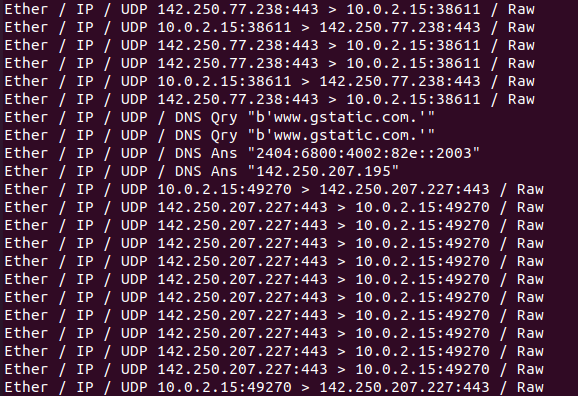
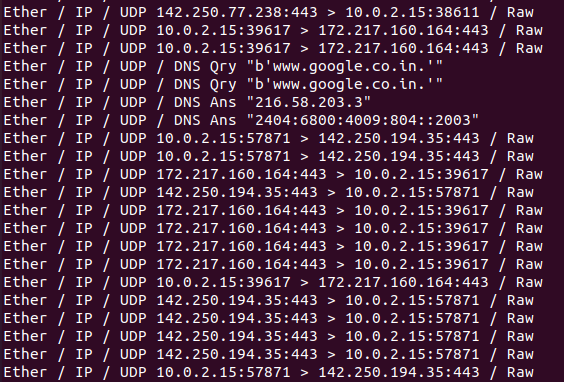


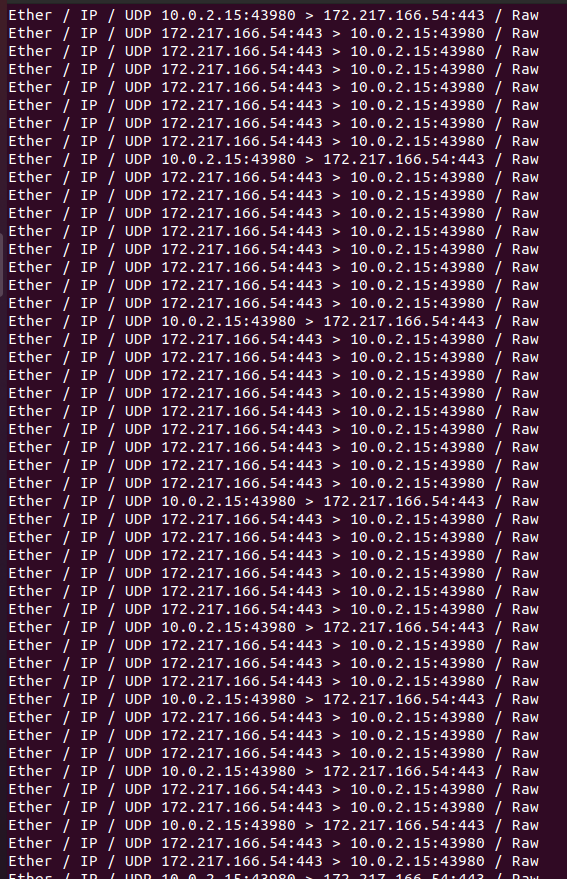
If we want continuous sniffing of traffic,we can replace the count with iface(interface) and we can monitor the network interface “enp0s3” used by Linux Ubuntu Machine and we can start seeing traffic and related information and we can refresh websites or access them and generate traffic and they can show us the continuously captured packets.They can show us the basics of the layered information which is very helpful.We canuse Ctrl+C to end the capturing and we can see the number of TCP,UDP,ICMP packets captured.If we want to use a more detailed view,we can use the count function to capture information on a single packet and instead of summary(),we can use the show() function to monitor that package.

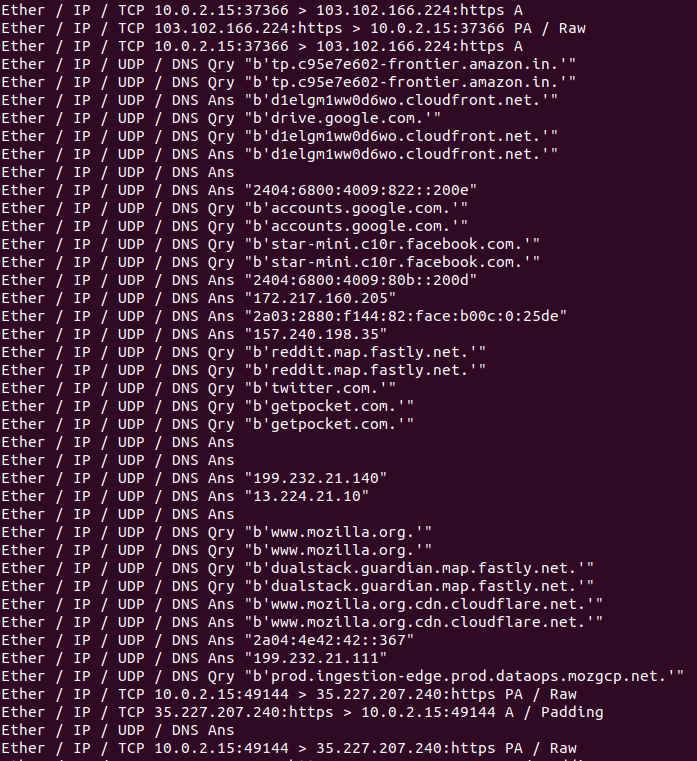


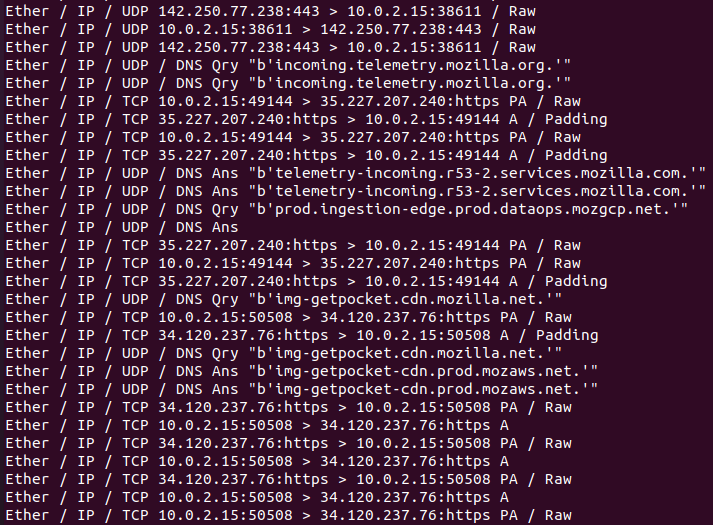
Accessing Google Search Engine:

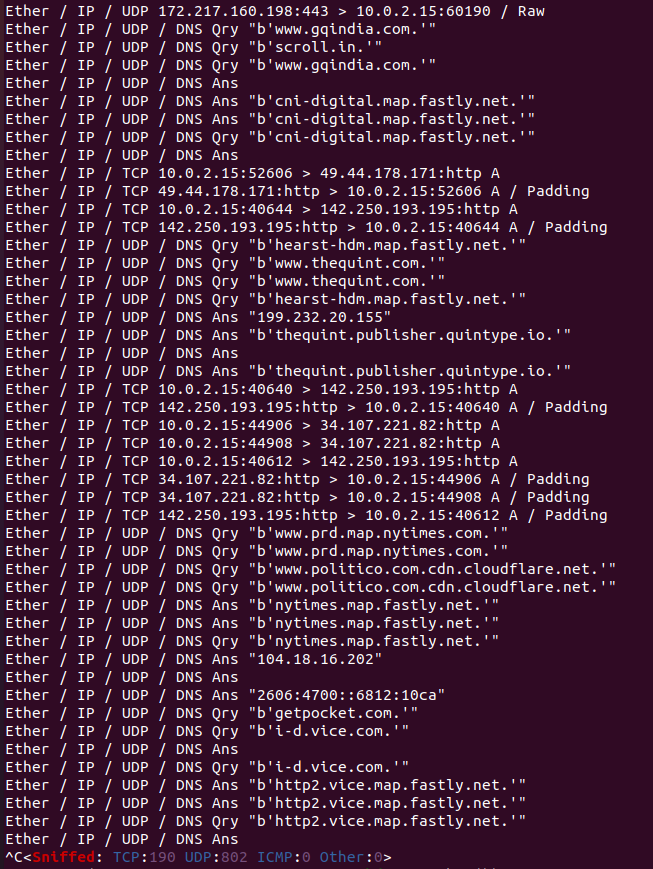




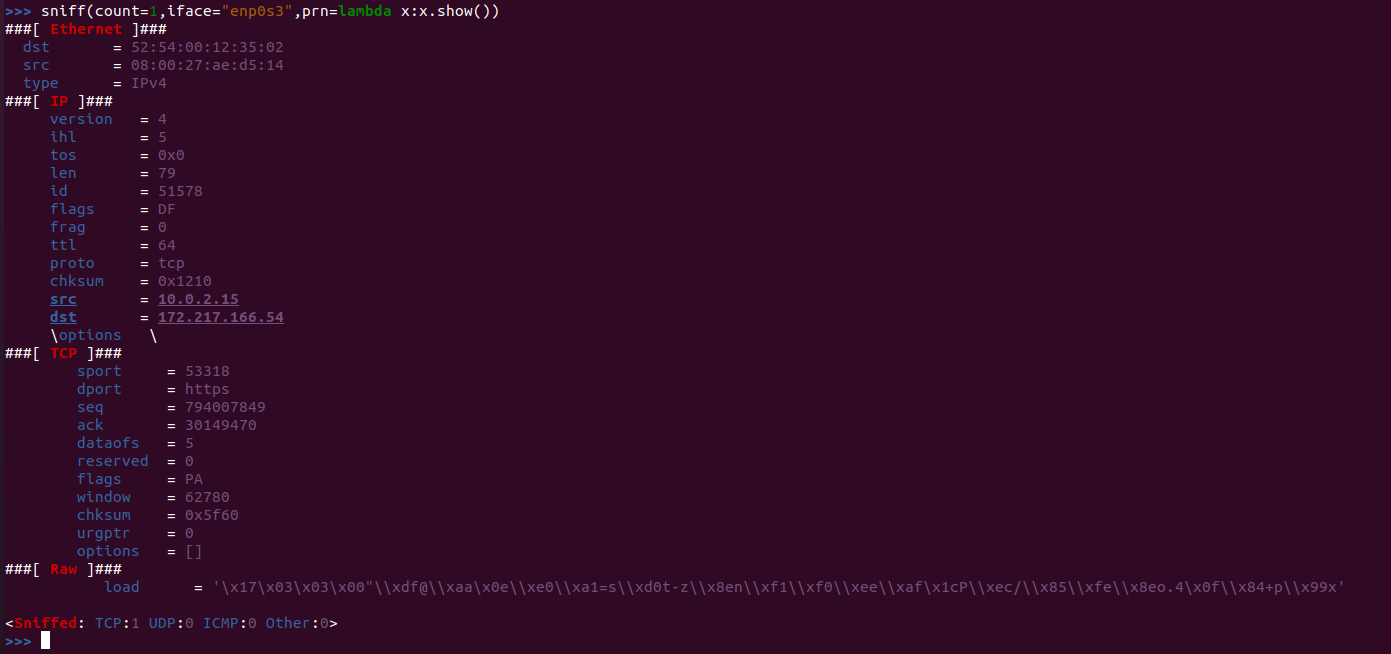
This is what shows in idle state during capturing: 

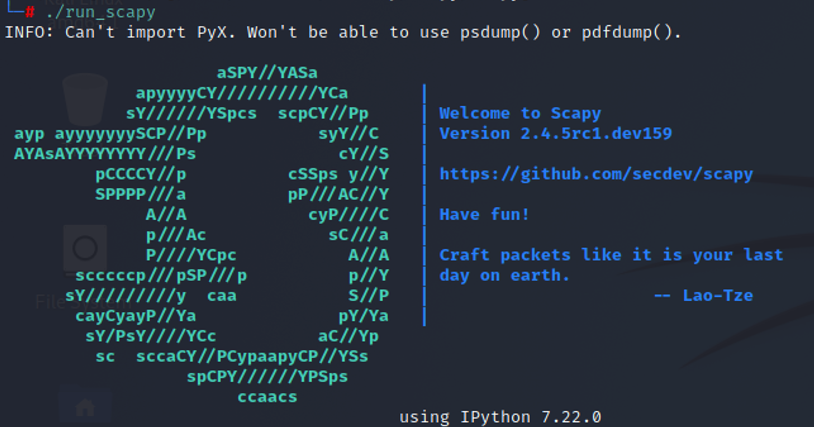


When we head back to Mozilla Firefox’s home page: 

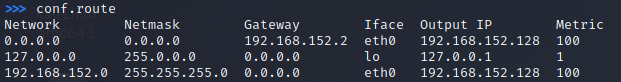
When the images and media of a site are loading up: 

Viewing details of a single captured packet using show() command:

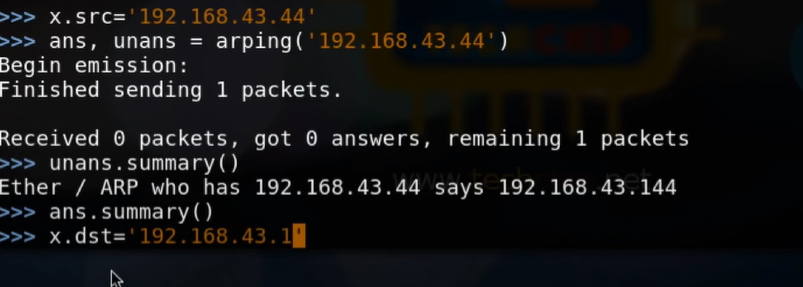


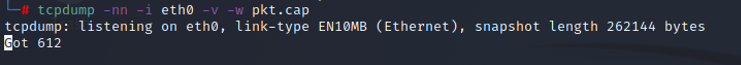


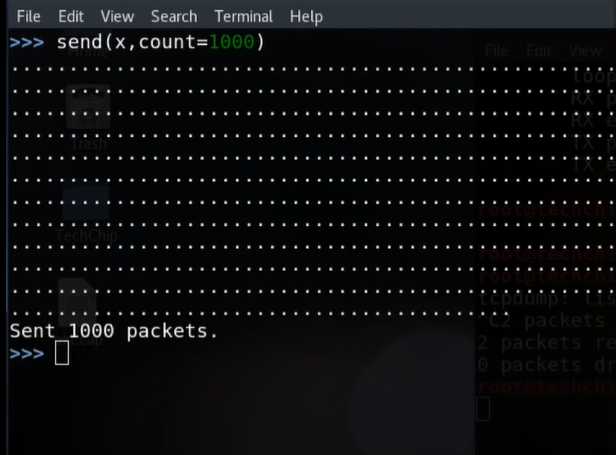
**Show the routing table of networks.**



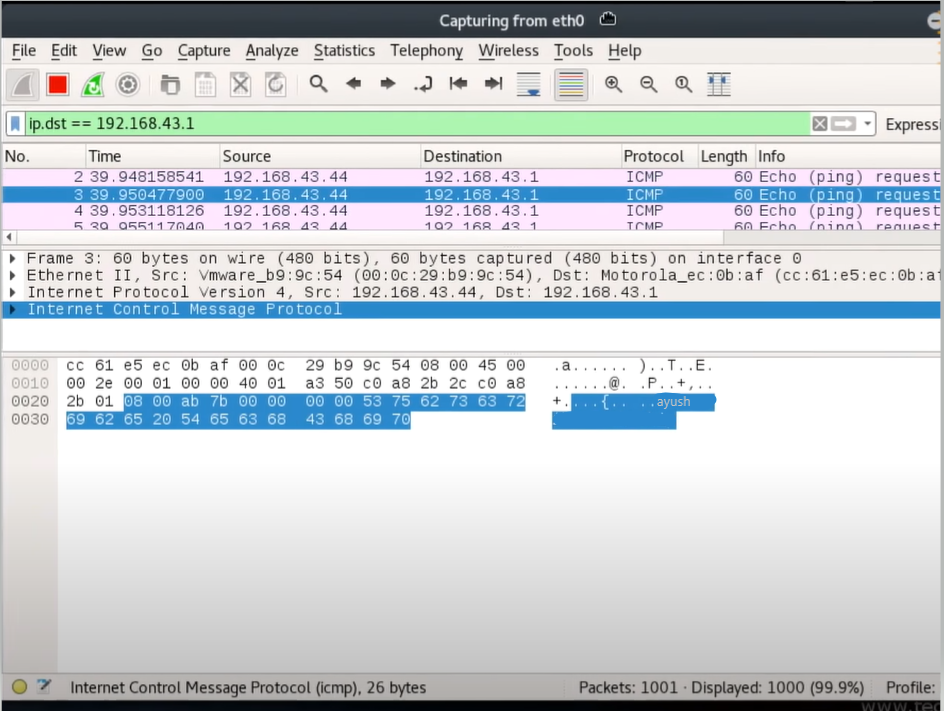
**TASK 2 Identify the subdomains**





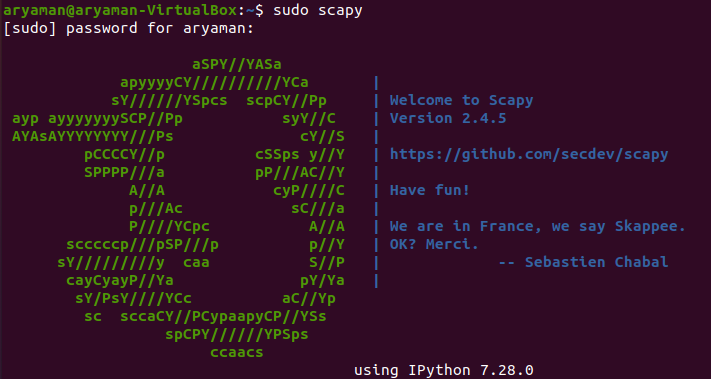


**TASK Analyze the file, i.e., find the no. of TCP, UDP packets etc, provide summary of it. Capturing packets sent using wireshark**

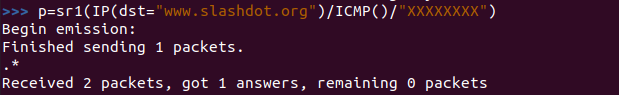


**Live packet sniffing using scapy**

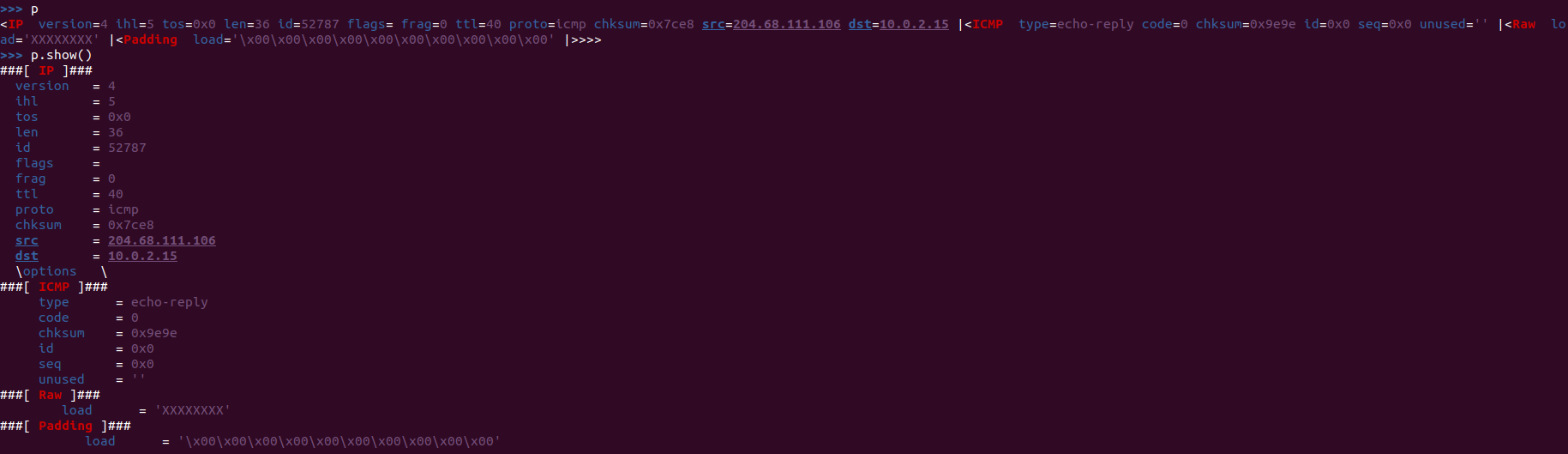
We will launch Scapy from our terminal:



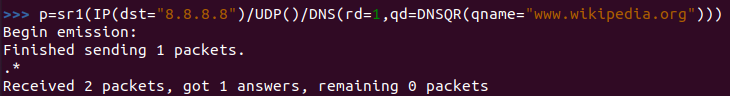
We will send and receive a single packet from Slashdot.org and will send an ICMP with a raw string in it.



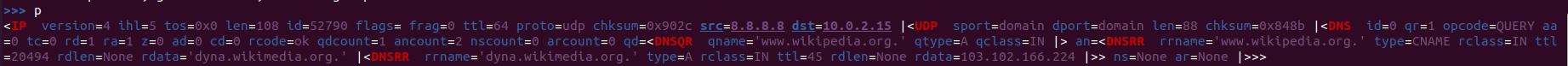
We can then monitor the details of the packets using show function.

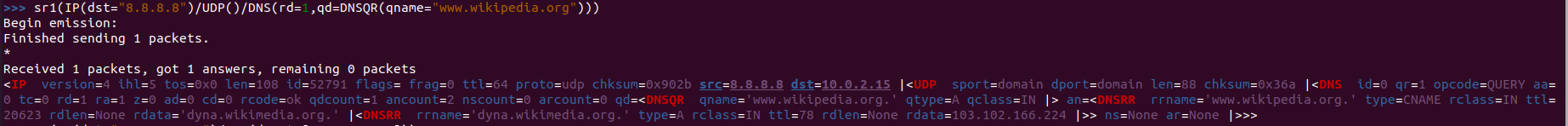


Packet has come back to us.We can use show command to view source address to the Linux Virutal Machine.We can see the echo replay and can view more details on the echo.



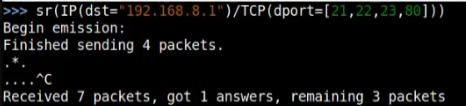
We will now perform a DNS Query and we will use Google’s DNS server.We will use UDP Protocol and we will put a DNS Query.RD=1 means recursive is desired and will get results recursively.We will then put our query name and will put in Wikipedia to resolve it.We will get DNS Query name and then the answer we will look for in our data is resource record name-Type A and we will get our IP address in rdata variable.The data was resolved between Google’s DNS server and my computer.





If we don’t want to put it into a variable,we can delete the assignment part and we can get the details from the send and receive command directly in the command line.

Now we will send and receive continuous packets.This time we will use only sr command to send and receive IP packets to destination(our gateway,my router) and we will send Destination port In which we will include 21,22,23,80.We will send and receive IP packets to my gateway we will use TCP for transfport and destination ports as mentioned above.We will send and receive 4 packets.



We got only 1 response from the routers of the web management interface.We will get the numbers of unanswered and answered packets.



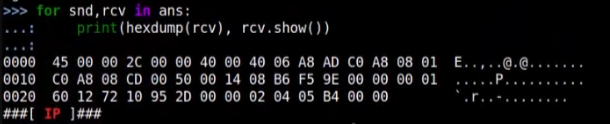
We will store the data received in ans,uans variables.

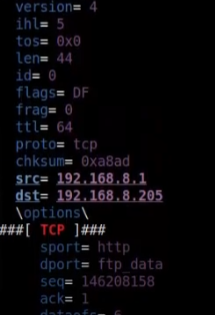
>>>ans,uans=\_

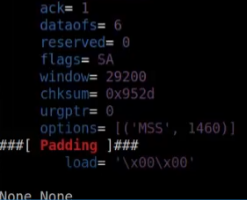


After we get the summary,we will notice that an answer has been sent from the gateway to us(with a SYN ACK).

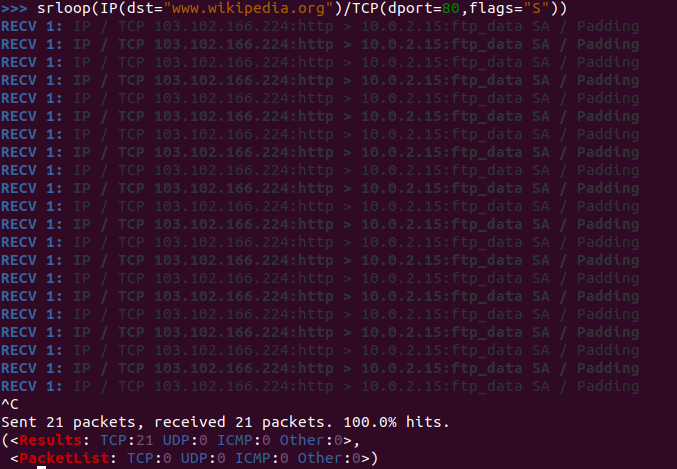
We will print hexdump of packets that we have received and the replies and we will loop through the answers.We want to print the fields of packets that we have received.Since it is SYN and an ACK,we will know that the machine was trying to finish the connection.

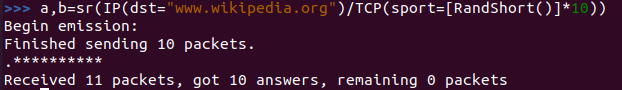




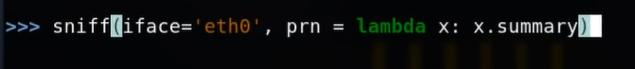


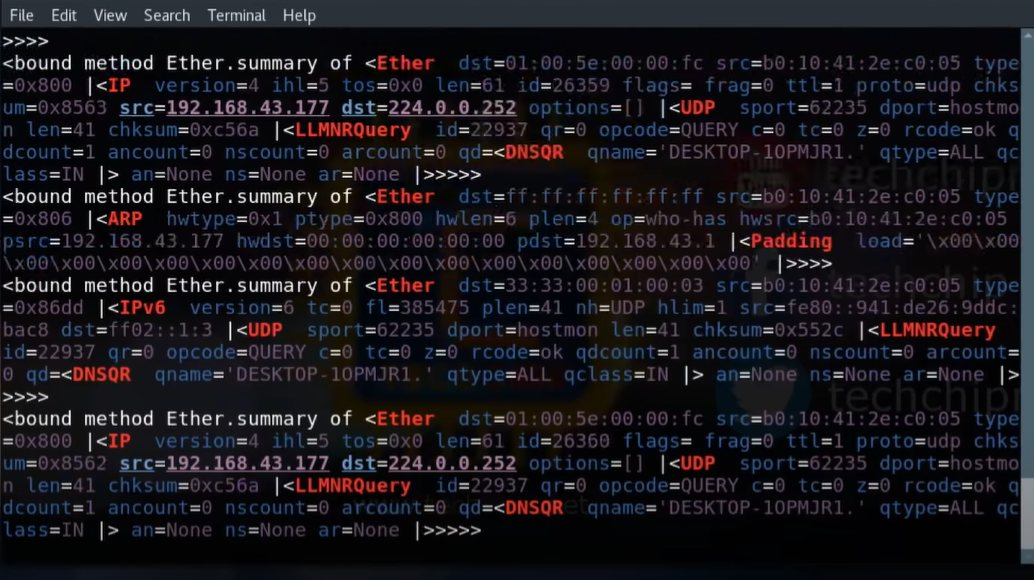
We can send and receive on a loop.Our destination would be Wikipedia and we will send a TCP-a destination port and we will also specify flags.In the TCP transmission to Wikipedia on a loop,we will contact Wikipedia on a loop and we will set the flag to SYN and we will perform a SYN flooding attack and we can specify the timing and how quickly it can happen.The SYN and the ACK will come back to us,which is basically a flooding attack against Wikipedia by sending repeated SYN requests-sending multiple times to establish a 3 way handshake.

  
We can change what we are sending and in the transport layer,we can specify the source port which will be random-it will look like 10 random sources are trying to establish a connection.



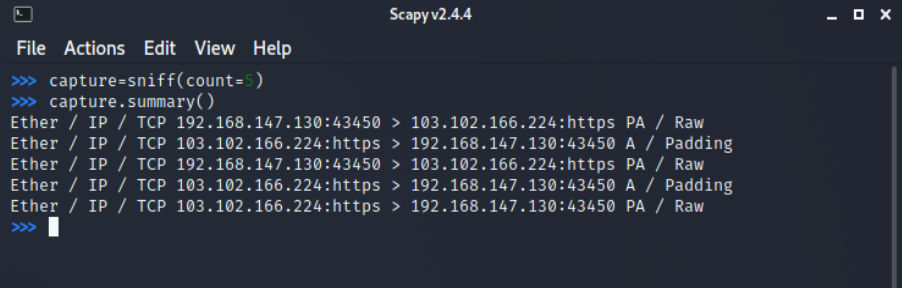
We can plot out our answers using a and b variable.We will plot it automatically using a lambda function and we can plot out our sent packets.We can plot our information from Scapy which is a powerful features and we can chart that information on our console as well.





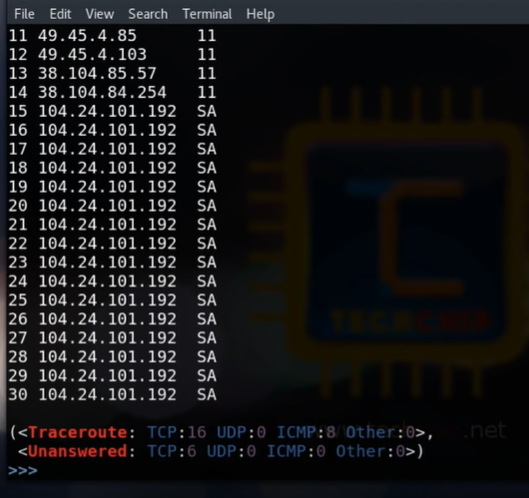
The sniff() function listens for an infinite period of time until the user interrupts.

To restrict the number of packets to be captured sniff() allows a count parameter. By specifying a value for the count, the packet capturing will be restricted to the specified number.



**TASK 4 Check the route of the provide URL using scapy.**



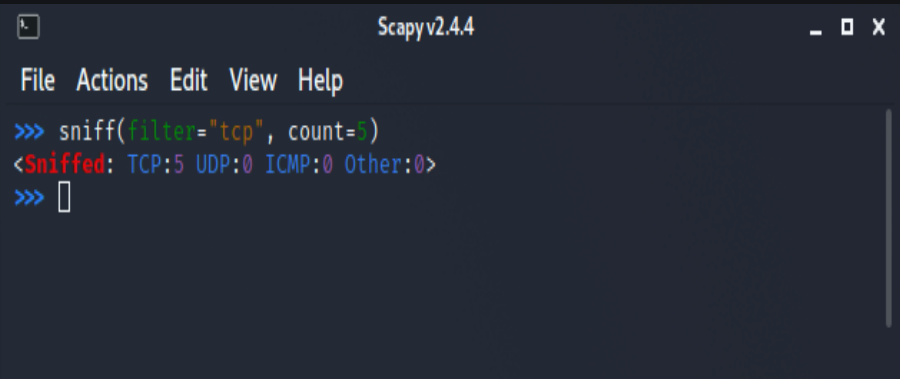


You can also filter packets while sniffing using the filter parameter. It uses a Berkeley Packet Filter (BPF) syntax.

The following command will capture only TCP packets:

sniff(filter="tcp", count=5)

Similarly, you can filter any packet on the basis of source/destination IP address, port number, protocol and lot more by using the BPF syntax.



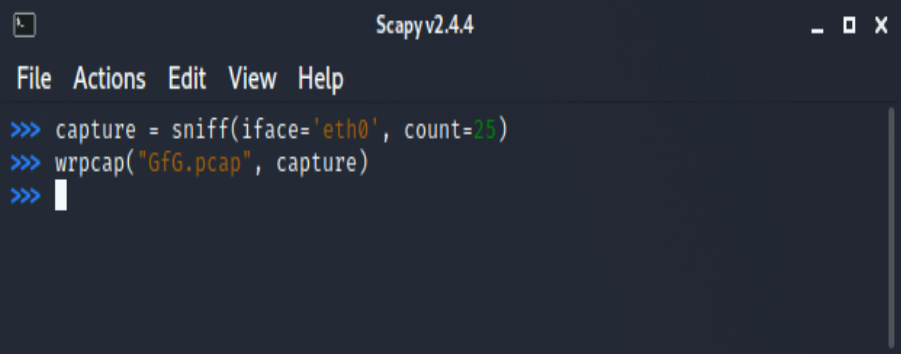
When scapy sniffs packets, it generally sniffs from all of your network interfaces. However, we can explicitly mention the interfaces that we would like to sniff on using the iface parameter. The iface can either be an element or a list of elements.

**TASK 5 Provide the packets sent and received summary using SCAPY, store the contents using PCAP file**

Scapy also allows us to store the sniffed packets in a pcap file. Running the following command will write the sniffed packets in a pcap:

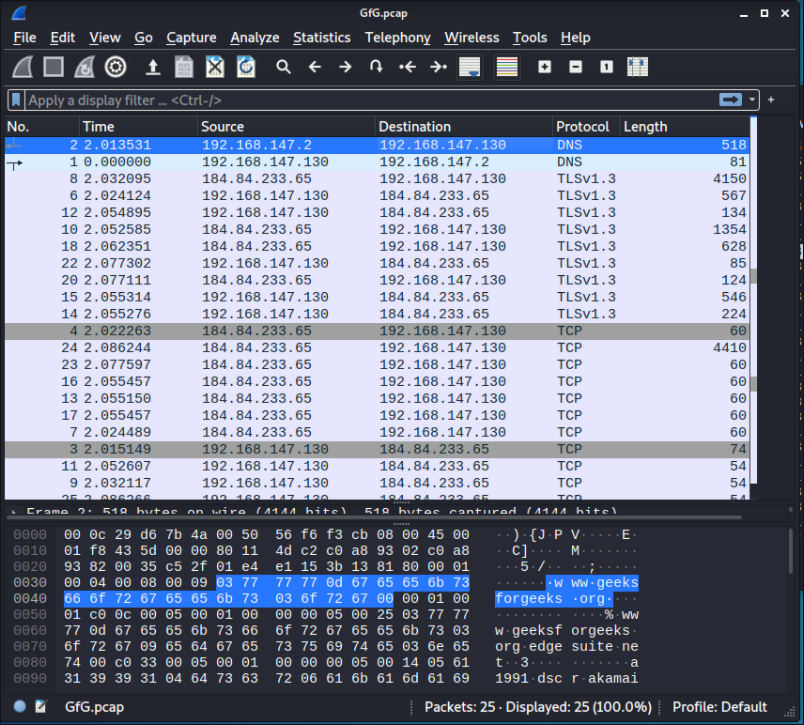
wrpcap("<file name>", capture)

where capture is the list of sniffed packets.



The stored pcap files can be analyzed using Wireshark, tcpdump, WinDump, Packet Square, etc.

Opening GfG.pcap using Wireshark:



We can also sniff packets offline from pcap files by running the following command:

sniff(offline="<file name>")

