Scapy is a powerful and versatile packet manipulation tool written in python. Using scapy, a user will be able send, sniff, dissect and forge network packets. Scapy also have the capability to store the sniffed packets in a pcap file. Using scapy, we will be able to handle tasks like tracerouting, probing, scanning, unit tests and network discovery with ease. All of these properties make scapy useful for network-based attacks.

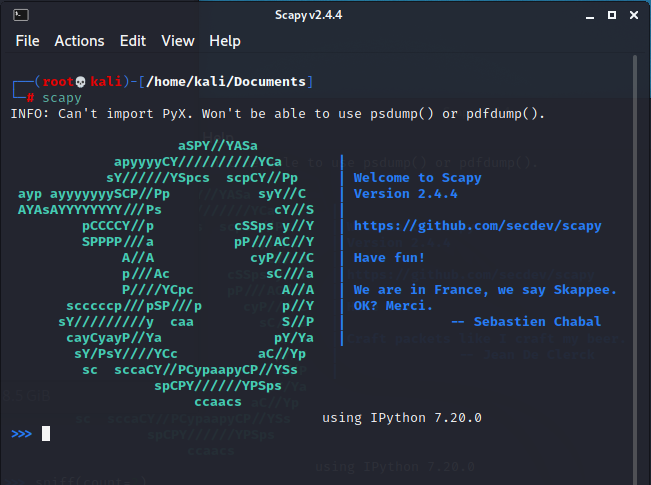
As mentioned before scapy performs a wide range of networking tasks and one such task is packet sniffing. **Packet sniffing** is a process of capturing all the packets flowing across a computer network. The sniffed packets give away a lot of information like what website does a user visit, what contents does the user see, what does the user download and almost everything. The captured packets are usually stored for future analysis.

In this article we will learn how to sniff packets using scapy and store the sniffed packets in a pcap file.

To work on with scapy, we need to have scapy installed in our computer. To install scapy run this command on your Linux terminal:

sudo apt-get install python3-scapy

Now that we have scapy installed, open scapy shell by typing “**scapy**” in your terminal.



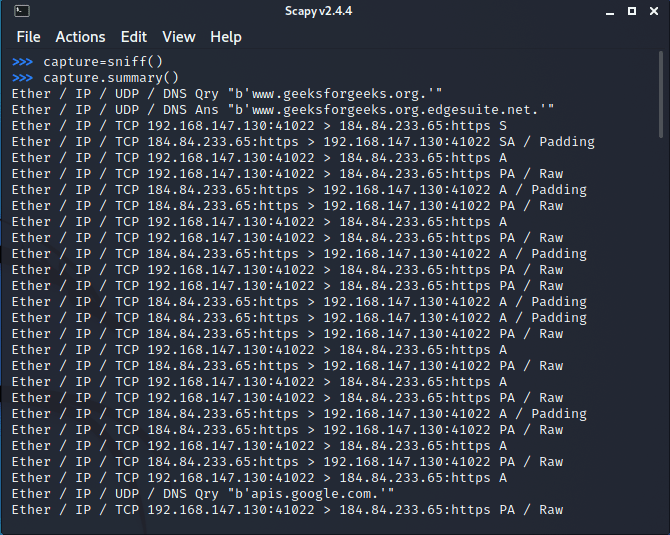
**Sniffing packets using scapy**

To sniff the packets use the **sniff()** function. The sniff() function returns information about all the packets that has been sniffed.

capture = sniff()

To see the summary of packet responses, use **summary()**

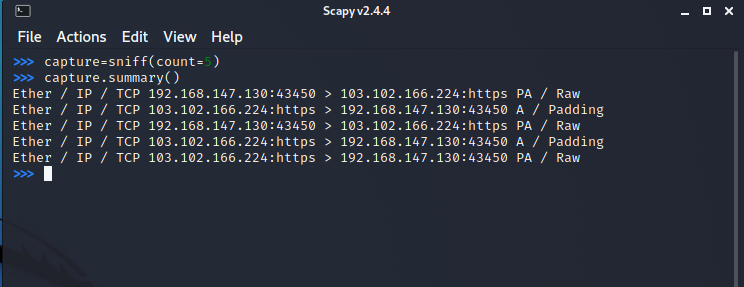
capture.summary()



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.

capture = sniff(count=5)

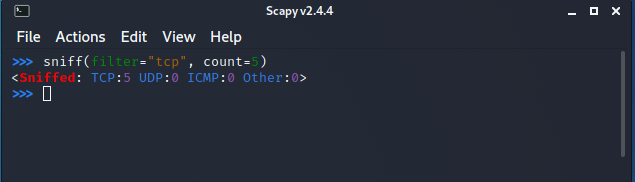


You can also filter packets while sniffing using the **filter** parameter. It uses a Berkeley Packet Filter (BPF) [syntax](https://biot.com/capstats/bpf.html).

The following command will capture only TCP packets:

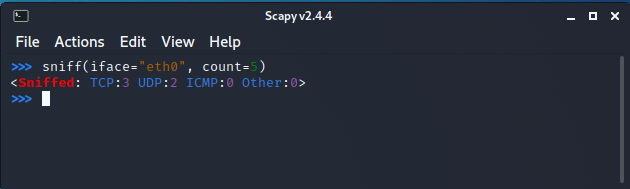
sniff(filter="tcp", count=5)

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.



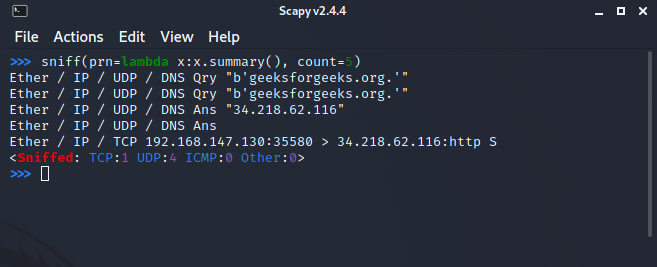
When scapy sniffs packet, it generally sniffs from all of your network interfaces. However, we can explicitly mention the interfaces that we would like to sniff on using the **iface** parameter. The iface can either be an element or a list of elements.

sniff(iface="eth0", count=5)



sniff() function has another interesting parameter called **prn** that allows you to pass a function that executes with each packet sniffed. This allows us to do some custom actions with each packet sniffed.

sniff(prn=lambda x:x.summary(), count=5)

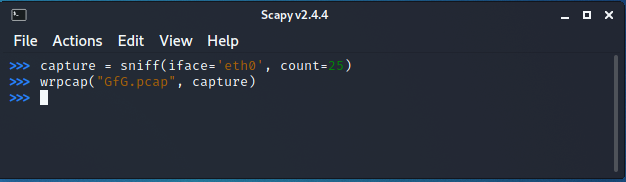


Scapy also allows us to store the sniffed packets in a **pcap** file.

Running the following command will write the sniffed packets in a pcap:

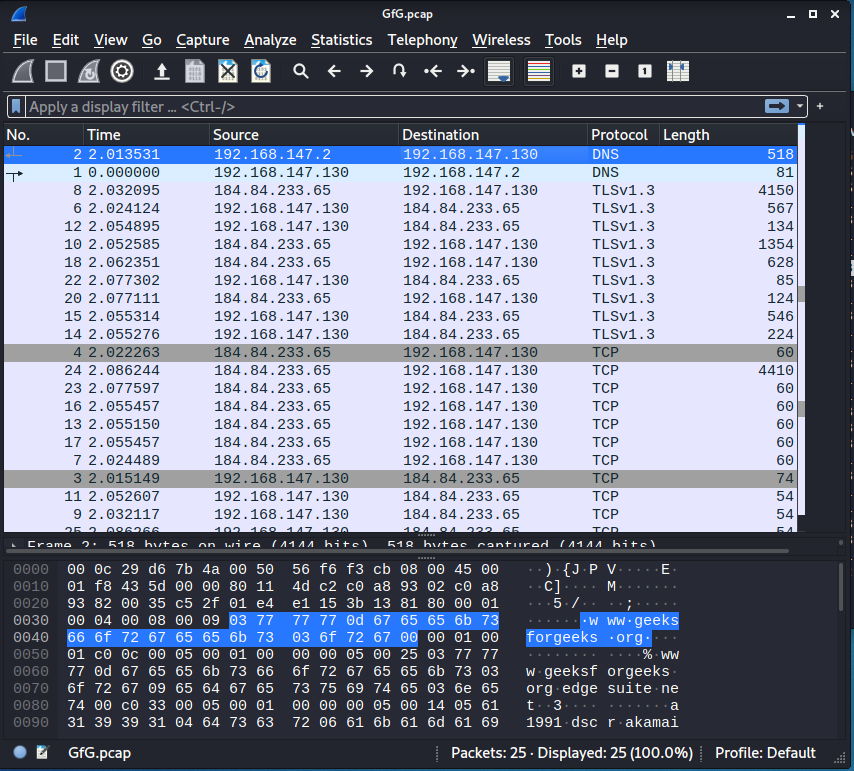
wrpcap("<file name>", capture)

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:**

Analyzing scapy sniffed packets in Wireshark

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

sniff(offline="<file name>")

