REPORT
on
"Network Sniffing with Wireshark Similar Tool"
Course: "Computer Security"

Course Code: CS4035D

Department of Computer Science and Engineering,
National Institute of Technology, Calicut
Calicut, Kerala, India – 673 601

Submitted by
Dr. Vasudevan, A. R.
Department of CSED

Submitted to
Abhijeet kumar singh
M190675CA
Ashish Kumar Sahu
M190365CA
Abhilasha Sharma
M180275CA

Chapter 1

Network Sniffing Using Wireshark

1.1 *Introduction*

1.1.1 What is Sniffing

Sniffing can be defined as a process that refers to investigating something covertly in order to find any confidential information. By an information security perspective, sniffing refers to tapping the traffic or routing the traffic to a target where it can be captured, analyzed and monitored. It is mostly performed to analyze the usage of network, troubleshooting any network issues, session monitoring for development and the testing purposes.

The network packets or TCP/IP packet contains vital information which is required for two network interfaces to communicate with each other. It contains fields that are source and destination IP addresses, ports, sequence numbers and the protocol type. All of these fields are crucial for various network layers to function properly, and especially for the Layer 7 application which makes use of the received data.

1.1.2 What is Packet Sniffing

The data that has to be transmitted over the computer network, is firstly and foremost broken down into smaller units at the sender's node called *data packets* and at the receiver's node it is resembled in original format. So we can say that it is the *smallest unit* of communication over a computer network. These packets are also called a block, a segment, a cell or a datagram. Capturing these data packets when it is flowing from one device to another over a network is known as **packet sniffing**. We can assume that it is similar to wiretapping to a telephone network.These are mostly used by *crackers and hackers* to collect information illegally from the network. *ISPs, advertisers and governments also use such softwares to capture packets.*

1.1.3 What is Wireshark

Wireshark is one of the best network packet analyzer, which is an essential tool for security professionals and system administrators. It is a free and open source packet analyzer, the purpose of this packet analyzer is to present captured packet data in as much detail as possible.

We can think that network packet analyzer is like a measuring device for examining what's happening inside a network cable,which is similar to an electrician that uses a voltmeter for examining what's happening inside an electric cable.It is a tool that intercepts traffic and converts that binary traffic into human-readable format, which makes it easy to identify what type of traffic is crossing your network, how much of it, how frequently it transfers, the latency there is between certain hops, and so forth.

1.1.4 ***References:***

https://www.greycampus.com/blog/information-security/what-is-a-sniffing-attack-and-how-can-you-defend%20it#: :text=From%20an%20information %20security%20perspective,for%20development%20and%20testing %20purpose.

https://www.valencynetworks.com/articles/cyber-security-attacks-network-sniffing.html

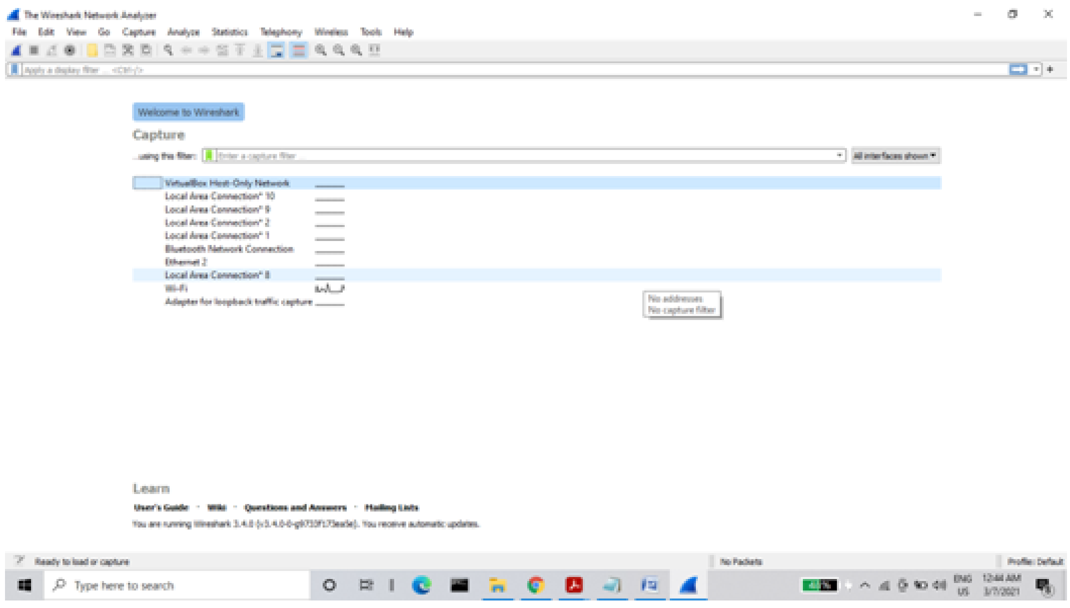
https://www.geeksforgeeks.org/what-is-packet-sniffing/

https://www.csoonline.com/article/3305805/what-is-wireshark-what-this-essential-troubleshooting-tool-does-and-how-to-use-it.html

Chapter 2

Working of wireshark

1) After successful installation of wireshark, launch the application. The interface of the application would be like

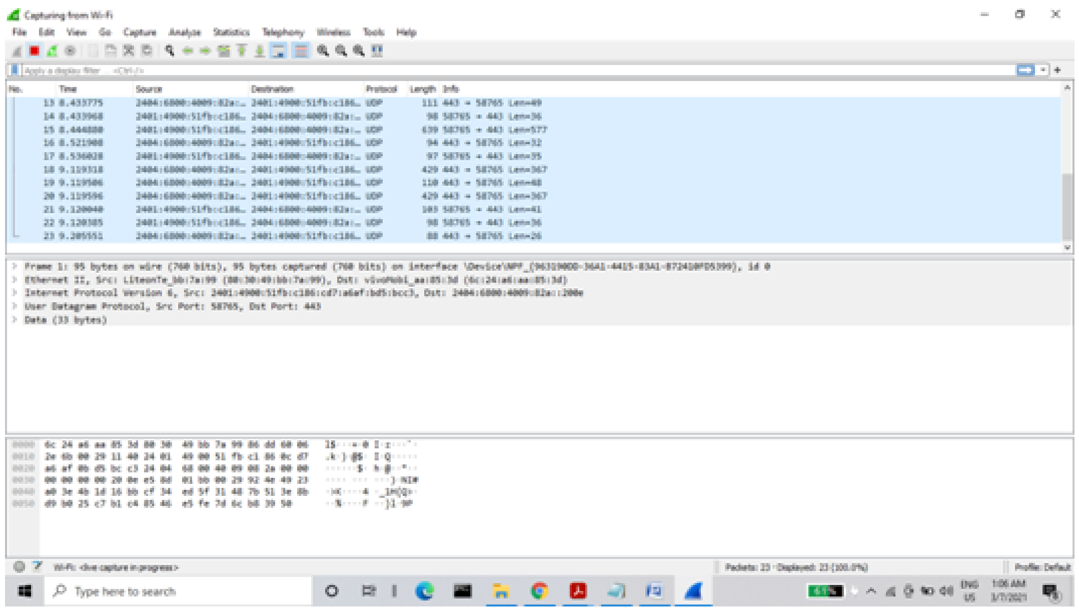


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
VirtualBox Host-Only Network
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 2
Local Area Connection* 1
Bluetooth Network Connection
Ethernet 2
Local Area Connection* 8
Wi-Fi
Adapter for loopback traffic capture
No addresses
No capture filter
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Ready to load or capture
No Packets
Profile: Default
Type here to search

You are running Wireshark 3.4.0 (v3.4.0-0-g9733f173ea5e). You receive automatic updates.

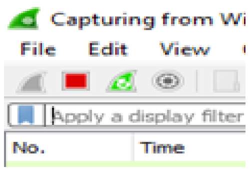
The picture above shows the homescreen of version 3.4.0 of the application.By selecting the current interface, we can get the traffic traversing through that interface.

2) The list in the above figure shows the Interface list options. That is the number of interfaces are present. The option that we select from the list will determine all the traffic we see. For instance, suppose we have selected the wi-fi option.the after this, a new window will open up, which shows the traffic that is flowing through the network currently. The picture below will show the live capture of packets from our wireshark.

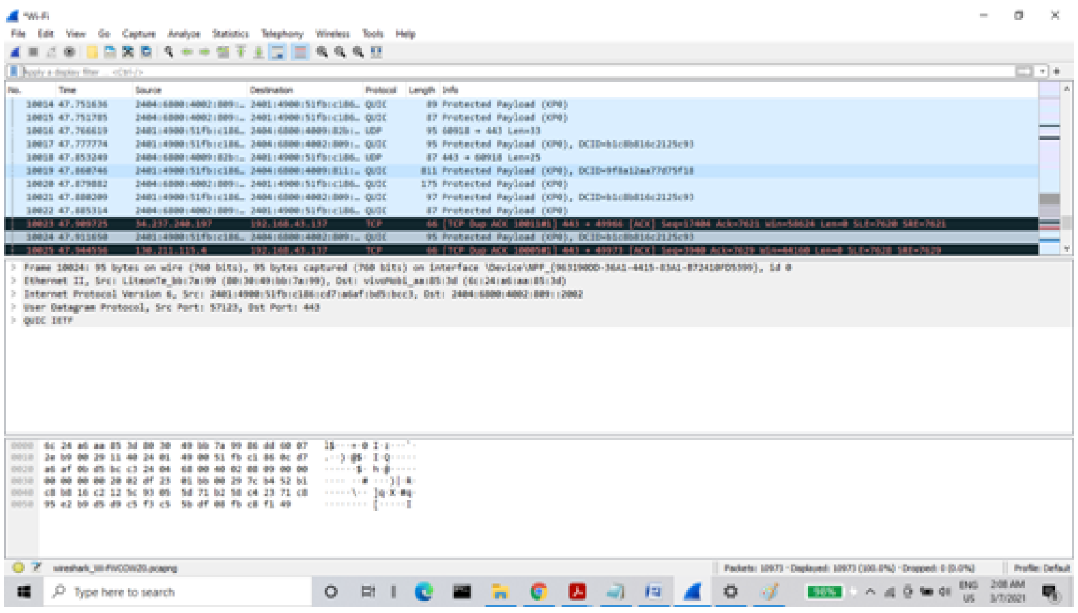


```
No.  Time      Source                    Destination               Protocol Length Info
  13 8.433775  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP      111 443 → 58765 Len=49
  14 8.433968  2401:4900:51fb:c186…      2404:6800:4009:82a:…      UDP       98 58765 → 443 Len=36
  15 8.444880  2401:4900:51fb:c186…      2404:6800:4009:82a:…      UDP      639 58765 → 443 Len=577
  16 8.521908  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP       94 443 → 58765 Len=32
  17 8.536028  2401:4900:51fb:c186…      2404:6800:4009:82a:…      UDP       97 58765 → 443 Len=35
  18 9.119318  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP      429 443 → 58765 Len=367
  19 9.119506  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP      110 443 → 58765 Len=48
  20 9.119596  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP      429 443 → 58765 Len=367
  21 9.120040  2401:4900:51fb:c186…      2404:6800:4009:82a:…      UDP      103 58765 → 443 Len=41
  22 9.120385  2401:4900:51fb:c186…      2404:6800:4009:82a:…      UDP       98 58765 → 443 Len=36
  23 9.205551  2404:6800:4009:82a:…      2401:4900:51fb:c186…      UDP       88 443 → 58765 Len=26

> Frame 1: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface \Device\NPF_{96319BDD-36A1-4415-83A1-B72410FD5399}, id 0
> Ethernet II, Src: LiteonTe_bb:7a:99 (80:30:49:bb:7a:99), Dst: vivoMobi_aa:85:3d (6c:24:a6:aa:85:3d)
> Internet Protocol Version 6, Src: 2401:4900:51fb:c186:cd7:a6af:bd5:bcc3, Dst: 2404:6800:4009:82a::200e
> User Datagram Protocol, Src Port: 58765, Dst Port: 443
> Data (33 bytes)

0000  6c 24 a6 aa 85 3d 80 30  49 bb 7a 99 86 dd 60 06
0010  2e 6b 00 29 11 40 24 01  49 00 51 fb c1 86 0c d7
0020  a6 af 0b d5 bc c3 24 04  68 00 40 09 08 2a 00 00
0030  00 00 00 00 20 0e e5 8d  01 bb 00 29 92 4e 49 23
0040  a0 3e 4b 1d 16 bb cf 34  ed 5f 31 48 7b 51 3e 8b
0050  d9 b0 25 c7 b1 c4 85 46  e5 fe 7d 6c b8 39 50

Wi-Fi: <live capture in progress>          Packets: 23 · Displayed: 23 (100.0%)          Profile: Default
```
Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Type here to search
1:06 AM
3/7/2021

3) Now your analyzer is capturing packets that are flowing through your network and to analyze any packet, click on the red button in menu bar. The packet captured until yet is present in second part "Packet listing window".It determines the packet flow or the captured packets in the traffic. It includes the packet number, time, source, destination, protocol, length, and info.



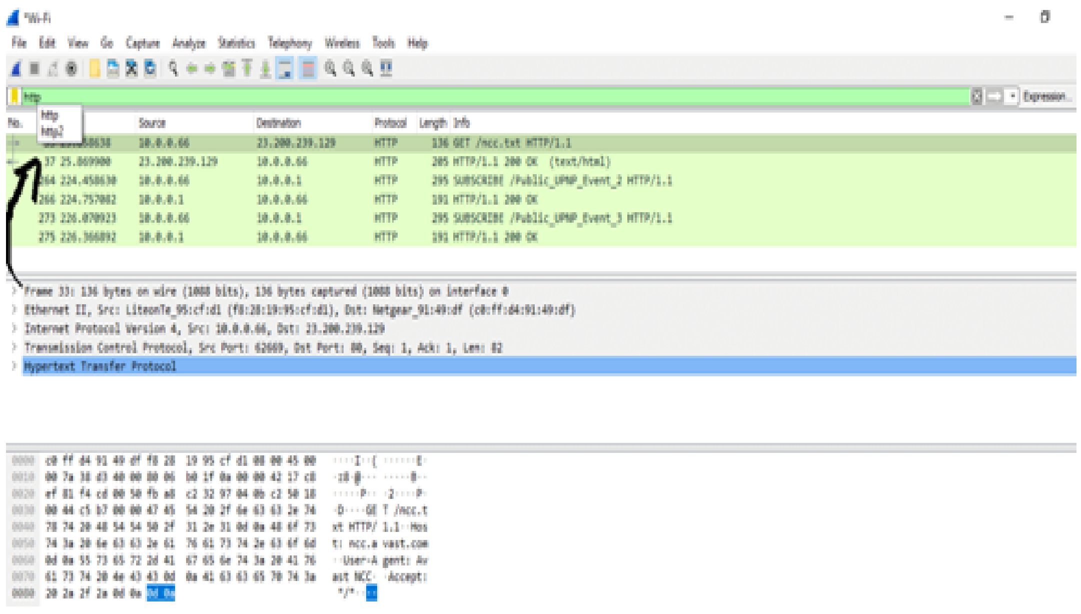
Below picture will show the list of captured packets



```
No.    Time       Source                 Destination            Protocol Length Info
10014 47.751636  2404:6800:4002:809:…   2401:4900:51fb:c186…   QUIC      89 Protected Payload (KP0)
10015 47.751785  2404:6800:4002:809:…   2401:4900:51fb:c186…   QUIC      87 Protected Payload (KP0)
10016 47.766619  2401:4900:51fb:c186…   2404:6800:4009:82b:…   UDP       95 60918 → 443 Len=33
10017 47.777774  2401:4900:51fb:c186…   2404:6800:4002:809:…   QUIC      95 Protected Payload (KP0), DCID=b1c8b816c2125c93
10018 47.853249  2404:6800:4009:82b:…   2401:4900:51fb:c186…   UDP       87 443 → 60918 Len=25
10019 47.860746  2401:4900:51fb:c186…   2404:6800:4009:811:…   QUIC     811 Protected Payload (KP0), DCID=9f8a12aa77d75f18
10020 47.879882  2404:6800:4002:809:…   2401:4900:51fb:c186…   QUIC     175 Protected Payload (KP0)
10021 47.888209  2401:4900:51fb:c186…   2404:6800:4002:809:…   QUIC      97 Protected Payload (KP0), DCID=b1c8b816c2125c93
10022 47.885314  2404:6800:4002:809:…   2401:4900:51fb:c186…   QUIC      87 Protected Payload (KP0)
10023 47.909725  34.237.240.197         192.168.43.137         TCP       66 [TCP Dup ACK 10011#1] 443 → 49966 [ACK] Seq=17404 Ack=7621 Win=50624 Len=0 SLE=7620 SRE=7621
10024 47.911650  2401:4900:51fb:c186…   2404:6800:4002:809:…   QUIC      95 Protected Payload (KP0), DCID=b1c8b816c2125c93
10025 47.944556  130.211.115.4          192.168.43.137         TCP       66 [TCP Dup ACK 10000#1] 443 → 49973 [ACK] Seq=3940 Ack=7029 Win=64160 Len=0 SLE=7028 SRE=7029

> Frame 10024: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface \Device\NPF_{96319BDD-36A1-4415-83A1-B72410FD5399}, id 0
> Ethernet II, Src: LiteonTe_bb:7a:99 (80:30:49:bb:7a:99), Dst: vivoMobi_aa:85:3d (6c:24:a6:aa:85:3d)
> Internet Protocol Version 6, Src: 2401:4900:51fb:c186:cd7:a6af:bd5:bcc3, Dst: 2404:6800:4002:809::2002
> User Datagram Protocol, Src Port: 57123, Dst Port: 443
> QUIC IETF

0000  6c 24 a6 aa 85 3d 80 30  49 bb 7a 99 86 dd 60 07
0010  2e b9 00 29 11 40 24 01  49 00 51 fb c1 86 0c d7
0020  a6 af 0b d5 bc c3 24 04  68 00 40 02 08 09 00 00
0030  00 00 00 00 20 02 df 23  01 bb 00 29 7c b4 52 b1
0040  c8 b0 16 c2 12 5c 93 05  5d 71 b2 58 c4 23 71 c8
0050  95 e2 b9 d5 d9 c5 f3 c5  5b df 08 fb c8 f1 49

wireshark_Wi-FiWCOW20.pcapng          Packets: 10973 · Displayed: 10973 (100.0%) · Dropped: 0 (0.0%)          Profile: Default
```
*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Type here to search
2:08 AM
3/7/2021

4) There is a filter block below the menu bar, from where a large amount of data can be filtered. For example, if we apply a filter for HTTP, only the interfaces with the HTTP will be listed. If you want to filter according to the source,

right-click on the source you want to filter and select 'Apply as Filter' and choose '...and filter.'


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
http
http2
Expression...

```
No.     Time          Source          Destination      Protocol  Length Info
    33  [illegible]8638  10.0.0.66     23.200.239.129   HTTP      136 GET /ncc.txt HTTP/1.1
    37  25.869900     23.200.239.129  10.0.0.66        HTTP      205 HTTP/1.1 200 OK  (text/html)
   264  224.458630    10.0.0.66       10.0.0.1         HTTP      295 SUBSCRIBE /Public_UPNP_Event_2 HTTP/1.1
   266  224.757082    10.0.0.1        10.0.0.66        HTTP      191 HTTP/1.1 200 OK
   273  226.070923    10.0.0.66       10.0.0.1         HTTP      295 SUBSCRIBE /Public_UPNP_Event_3 HTTP/1.1
   275  226.366892    10.0.0.1        10.0.0.66        HTTP      191 HTTP/1.1 200 OK
```

```
> Frame 33: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits) on interface 0
> Ethernet II, Src: LiteonTe_95:cf:d1 (f8:28:19:95:cf:d1), Dst: Netgear_91:49:df (c0:ff:d4:91:49:df)
> Internet Protocol Version 4, Src: 10.0.0.66, Dst: 23.200.239.129
> Transmission Control Protocol, Src Port: 62669, Dst Port: 80, Seq: 1, Ack: 1, Len: 82
> Hypertext Transfer Protocol
```

```
0000  c0 ff d4 91 49 df f8 28  19 95 cf d1 08 00 45 00   ····I··( ······E·
0010  00 7a 38 d3 40 00 80 06  b0 1f 0a 00 00 42 17 c8   ·z8·@··· ·····B··
0020  ef 81 f4 cd 00 50 fb a8  c2 32 97 04 0b c2 50 18   ·····P·· ·2····P·
0030  00 44 c5 b7 00 00 47 45  54 20 2f 6e 63 63 2e 74   ·D····GE T /ncc.t
0040  78 74 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   xt HTTP/ 1.1··Hos
0050  74 3a 20 6e 63 63 2e 61  76 61 73 74 2e 63 6f 6d   t: ncc.a vast.com
0060  0d 0a 55 73 65 72 2d 41  67 65 6e 74 3a 20 41 76   ··User-A gent: Av
0070  61 73 74 20 4e 43 43 0d  0a 41 63 63 65 70 74 3a   ast NCC· ·Accept:
0080  20 2a 2f 2a 0d 0a 0d 0a                             */*····
```

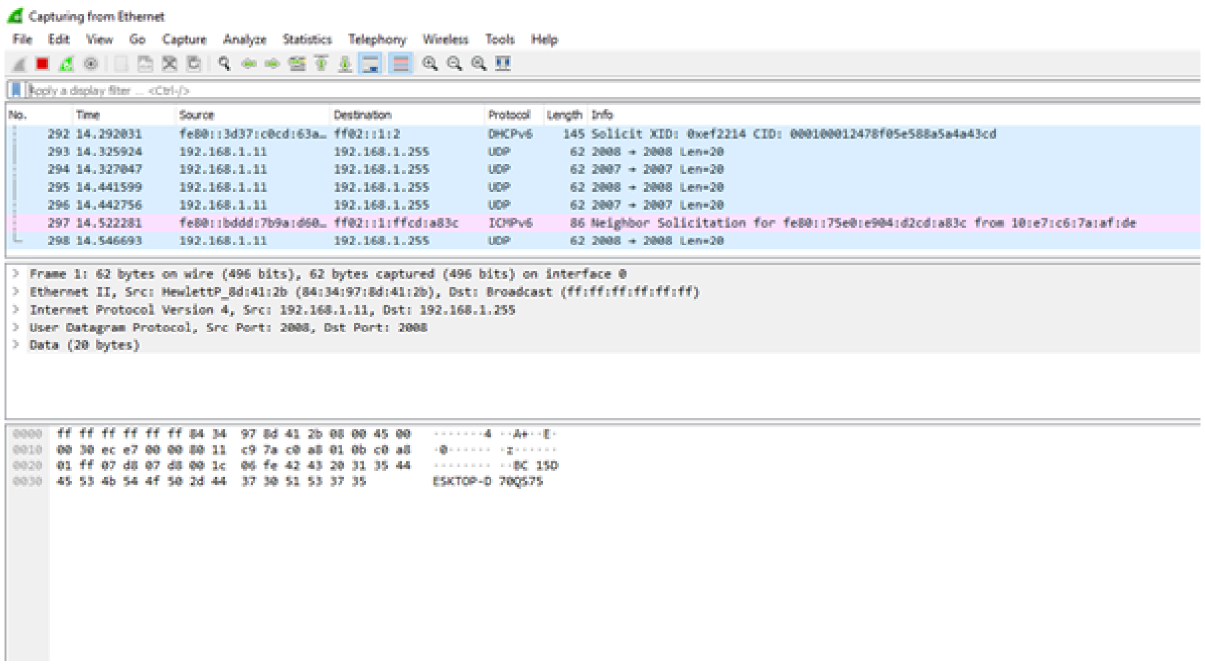
Chapter 3

Wireshark Packet Sniffing

It is a software that is mostly used by system administrators to isolate and troubleshoot the problems over a network. It is also used by hackers to analyze the packet of victims so that they can find any vulnerability of the system and used it to exploit their victims. It can also be used to capture any username and password that is transmitting without any encrypted connection. The wrong way of its used is to drop user confidential data.

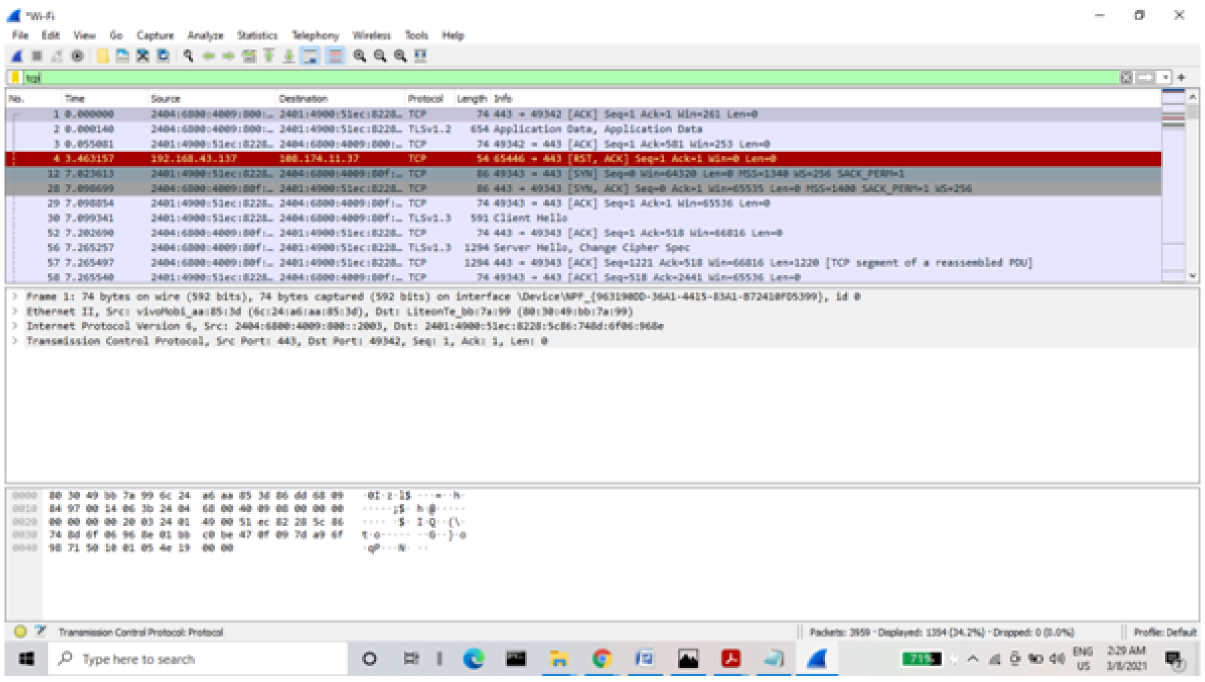
Given below is the steps for packet sniffing over a network:

- Start your Wireshark Application.
- Select the current interface. For example, the interface can be Ethernet that we would be using.
- Now a continuous flow of traffic can be shown on our screen. To stop or watch any particular packet and want to analyze any packet, we can press the red button below the menu bar, it will stop capturing the packet and now we can analyze all the previously captured packets.


Capturing from Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time        Source                 Destination          Protocol  Length  Info
  292 14.292031   fe80::3d37:c0cd:63a…   ff02::1:2            DHCPv6    145 Solicit XID: 0xef2214 CID: 000100012478f05e588a5a4a43cd
  293 14.325924   192.168.1.11           192.168.1.255        UDP        62 2008 → 2008 Len=20
  294 14.327047   192.168.1.11           192.168.1.255        UDP        62 2007 → 2007 Len=20
  295 14.441599   192.168.1.11           192.168.1.255        UDP        62 2008 → 2008 Len=20
  296 14.442756   192.168.1.11           192.168.1.255        UDP        62 2007 → 2007 Len=20
  297 14.522281   fe80::bddd:7b9a:d60…   ff02::1:ffcd:a83c    ICMPv6     86 Neighbor Solicitation for fe80::75e0:e904:d2cd:a83c from 10:e7:c6:7a:af:de
  298 14.546693   192.168.1.11           192.168.1.255        UDP        62 2008 → 2008 Len=20

> Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
> Ethernet II, Src: HewlettP_8d:41:2b (84:34:97:8d:41:2b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.255
> User Datagram Protocol, Src Port: 2008, Dst Port: 2008
> Data (20 bytes)

0000  ff ff ff ff ff ff 84 34  97 8d 41 2b 08 00 45 00   ·······4 ··A+··E·
0010  00 30 ec e7 00 00 80 11  c9 7a c0 a8 01 0b c0 a8   ·0······ ·z······
0020  01 ff 07 d8 07 d8 00 1c  06 fe 42 43 20 31 35 44   ········ ··BC 15D
0030  45 53 4b 54 4f 50 2d 44  37 30 51 53 37 35         ESKTOP-D 70QS75
```

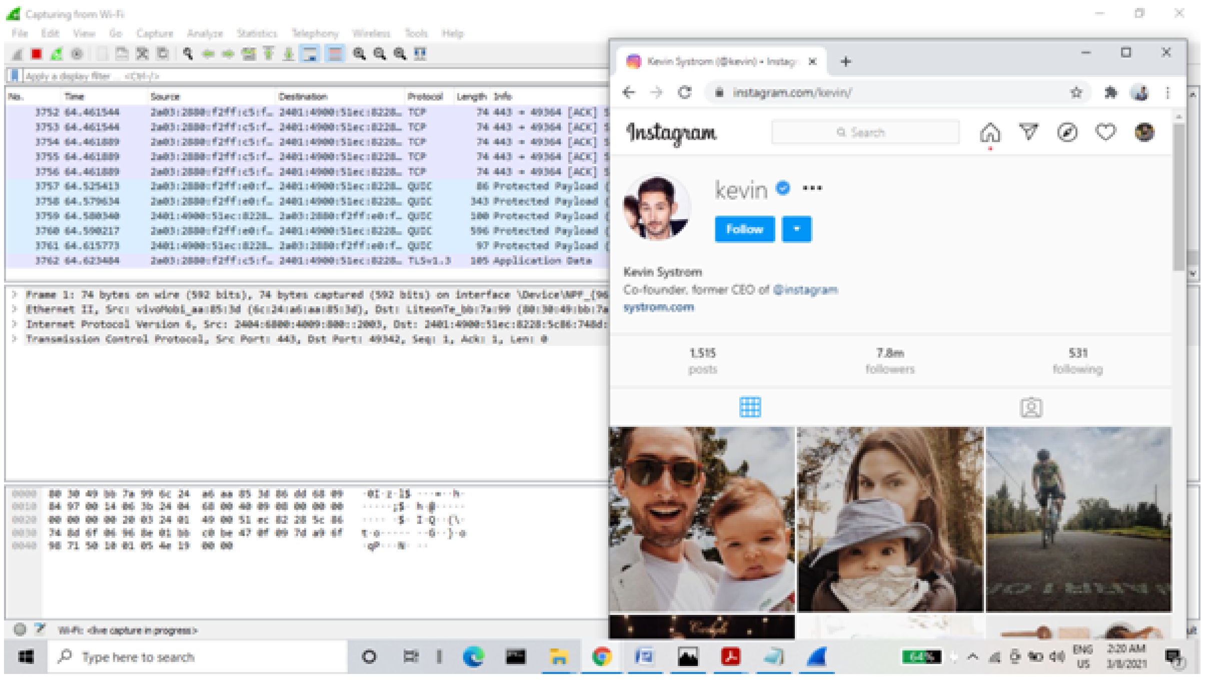
Apply the filter by the name 'tcp.' After the filter is applied, the screen will look as:



```
tcp
No.  Time      Source                  Destination             Protocol Length Info
   1 0.000000  2404:6800:4009:800:…  2401:4900:51ec:8228…  TCP        74 443 → 49342 [ACK] Seq=1 Ack=1 Win=261 Len=0
   2 0.000140  2404:6800:4009:800:…  2401:4900:51ec:8228…  TLSv1.2   654 Application Data, Application Data
   3 0.055081  2401:4900:51ec:8228…  2404:6800:4009:800:…  TCP        74 49342 → 443 [ACK] Seq=1 Ack=581 Win=253 Len=0
   4 3.463157  192.168.43.137          108.174.11.37           TCP        54 65446 → 443 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  12 7.023613  2401:4900:51ec:8228…  2404:6800:4009:80f:…  TCP        86 49343 → 443 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256 SACK_PERM=1
  28 7.098699  2404:6800:4009:80f:…  2401:4900:51ec:8228…  TCP        86 443 → 49343 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=256
  29 7.098854  2401:4900:51ec:8228…  2404:6800:4009:80f:…  TCP        74 49343 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  30 7.099341  2401:4900:51ec:8228…  2404:6800:4009:80f:…  TLSv1.3   591 Client Hello
  52 7.202690  2404:6800:4009:80f:…  2401:4900:51ec:8228…  TCP        74 443 → 49343 [ACK] Seq=1 Ack=518 Win=66816 Len=0
  56 7.265257  2404:6800:4009:80f:…  2401:4900:51ec:8228…  TLSv1.3  1294 Server Hello, Change Cipher Spec
  57 7.265497  2404:6800:4009:80f:…  2401:4900:51ec:8228…  TCP      1294 443 → 49343 [ACK] Seq=1221 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
  58 7.265540  2401:4900:51ec:8228…  2404:6800:4009:80f:…  TCP        74 49343 → 443 [ACK] Seq=518 Ack=2441 Win=65536 Len=0

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{96319BDD-36A1-4415-83A1-872410FD5399}, id 0
> Ethernet II, Src: vivoMobi_aa:85:3d (6c:24:a6:aa:85:3d), Dst: LiteonTe_bb:7a:99 (80:30:49:bb:7a:99)
> Internet Protocol Version 6, Src: 2404:6800:4009:800::2003, Dst: 2401:4900:51ec:8228:5c86:748d:6f06:968e
> Transmission Control Protocol, Src Port: 443, Dst Port: 49342, Seq: 1, Ack: 1, Len: 0

0000  80 30 49 bb 7a 99 6c 24  a6 aa 85 3d 86 dd 68 09   ·0I·z·l$ ···=··h·
0010  84 97 00 14 06 3b 24 04  68 00 40 09 08 00 00 00   ·····;$· h·@·····
0020  00 00 00 00 20 03 24 01  49 00 51 ec 82 28 5c 86   ···· ·$· I·Q··(\·
0030  74 8d 6f 06 96 8e 01 bb  c0 be 47 0f 09 7d a9 6f   t·o····· ··G··}·o
0040  98 71 50 10 01 05 4e 19  00 00                     ·qP···N· ··

Transmission Control Protocol: Protocol      Packets: 3959 · Displayed: 1354 (34.2%) · Dropped: 0 (0.0%)      Profile: Default
```

The above picture is showing all the traffic which follows ‘tcp’ protocol.

In the below picture we can see that it is capturing packets that are flowing through this device,and wireshark is showing whole lots of information about the packet.



```
Capturing from Wi-Fi
Apply a display filter … <Ctrl-/>
No.  Time       Source                 Destination            Protocol Length Info
3752 64.461544  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TCP       74 443 → 49364 [ACK]
3753 64.461544  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TCP       74 443 → 49364 [ACK]
3754 64.461889  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TCP       74 443 → 49364 [ACK]
3755 64.461889  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TCP       74 443 → 49364 [ACK]
3756 64.461889  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TCP       74 443 → 49364 [ACK]
3757 64.525413  2a03:2880:f2ff:e0:f…  2401:4900:51ec:8228…  QUIC      86 Protected Payload
3758 64.579634  2a03:2880:f2ff:e0:f…  2401:4900:51ec:8228…  QUIC     343 Protected Payload
3759 64.580340  2401:4900:51ec:8228…  2a03:2880:f2ff:e0:f…  QUIC     100 Protected Payload
3760 64.590217  2a03:2880:f2ff:e0:f…  2401:4900:51ec:8228…  QUIC     596 Protected Payload
3761 64.615773  2401:4900:51ec:8228…  2a03:2880:f2ff:e0:f…  QUIC      97 Protected Payload
3762 64.623484  2a03:2880:f2ff:c5:f…  2401:4900:51ec:8228…  TLSv1.3  105 Application Data

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{96
> Ethernet II, Src: vivoMobi_aa:85:3d (6c:24:a6:aa:85:3d), Dst: LiteonTe_bb:7a:99 (80:30:49:bb:7a
> Internet Protocol Version 6, Src: 2404:6800:4009:800::2003, Dst: 2401:4900:51ec:8228:5c86:748d:
> Transmission Control Protocol, Src Port: 443, Dst Port: 49342, Seq: 1, Ack: 1, Len: 0

Wi-Fi: <live capture in progress>
```

This process of capturing the packet and analyzing is called as “Packet Sniffing”.

Chapter 4

Implementation of tool that captures Network packets like Wireshark

4.0.1 Libraries required to capture the packets

There are some specific libraries that are present in python library to implement the process of capturing the packets that are flowing through the network.

- Scapy

  It is used for large scale web scraping, Scrapy is actually a python framework. It provides us all the tools that we need to efficiently ***extract*** data from websites, and then process them as we want, after that store them in the preferred ***structure*** and format.

  As we all know the diverse nature of the internet, so there are no "one size fits all" approach in extracting data from any of the websites. Most of the time an ad hoc approach is taken and if we begin with writing a code for every little task you perform, we would eventually end up creating your own scraping framework. So there comes the Scrapy framework.

- Libpacp

  libpcap is a lightweight Python package, based on the *ctypes* library.

  It is fully compliant implementation of the original C *libpcap* from 1.0.0 up to 1.9.0 API and the *WinPcap*'s 4.1.3 libpcap (1.0.0rel0b) API by implementing whole its functionality in a clean Python instead of C.

- Libtrace

  python-libtrace is not a complete set of all the libtrace routines, rather it's a somewhat simplified set, intended as an easy-to-use toolkit, and one that should be suitable for networking students. It use (more or less) the original field names within the various header decodes.

- Socket Packet Capture

A simple packet sniffer in Python can be created with the help socket module. We can use the raw

socket type to get the packets. A raw socket provides access to the underlying protocols, which support

socket abstractions. Since raw sockets are part of the internet socket API, they can only be used to

generate and receive IP packets.

The processing power used is comparatively lower than the existing packet capturing libraries

Chapter 5

Methodology used for the preparation of Project

5.0.1 Methodology Used

The Technique and Methodologies that we consider for the development for this project that is “Network Packet Sniffer Tool Like Wireshark” is rational unified process and evolutionary prototyping model.

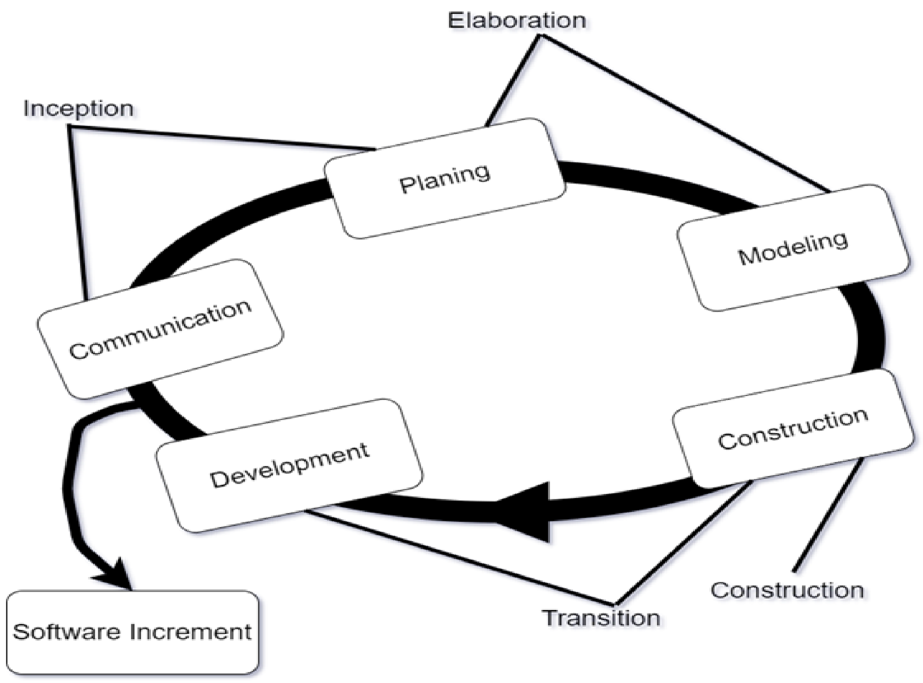


Figure: Rational Unified process: Flow of the work

The Rational Unified Process is a kind of comprehensive process framework that provides industry-tested practices for software and systems delivery and implementation and for effective project management. It is one of many processes contained within the Rational Process Library.

Rational Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer’s view of a system. It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development (Pressman, 2010).

5.0.2 Software Dependencies:

For the effective implementation/initialization of the project i.e. Network sniffing with wireshark similar tool requires a proper execution environment to be considered.

Python2.7 Programming Language accompanied with important libraries such as STRUCT module, STRUCT module and JSON module.

For the proper implementation of our project, the most important dependency of the project is the script has to be run with administrative privileges.

5.0.3 Hardware Dependencies:

The major hardware dependencies contingent for the initialization of the project are as follows:

- The System is dependent upon Mouse application.
- The System is dependent upon Monitor application
- Network Interface card is must for process of packet capture.
- 3-5 MB of hard disk space.
- 1 GB of RAM (Random Access Memory)

5.0.4 *References:*

https://www.javatpoint.com/wireshark

https://www.wireshark.org/docs/wsdg_html_chunked/PartDevelopment.html

https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/#: :text=Scrapy%20is%20a%20Python%20framework,your %20preferred%20structure%20and%20format.

https://pypi.org/project/libpcap/#: :text=libpcap%20is%20a%20lightweight %20Python,API%20and%20the%20WinPcap's%204.1

Chapter 6

Implementation of Packet Sniffer

6.1 Code for capturing packet and store in a “.pcap” file:

Implemented by Abhilasha Sharma

```
        # Create socket
        if os.name == "nt":
            conn = socket.socket(socket.AF_INET,socket.
               SOCK_RAW,socket.IPPROTO_IP)
            conn.bind((input("[+] YOUR_INTERFACE : "),0))
            conn.setsockopt(socket.IPPROTO_IP,socket.
               IP_HDRINCL,1)
            conn.ioctl(socket.SIO_RCVALL,socket.RCVALL_ON)
        else:
            conn = socket.socket(socket.AF_PACKET, socket.
               SOCK_RAW, socket.ntohs(0x0800))

        # datetime object containing current date and time

        #now = datetime.now()
        count=0

        rounds = int(input("\n\nEnter Number of Packets you
           want to capture: "))

        # dd/mm/YY H:M:S
        dt_string = time.strftime("%d-%m-%Y-%H:%M:%S")

        pcap_file1 = Pcap("Sniffed_packet_"+dt_string+".pcap")

        while True:

            count+=1

            raw_data, addr = conn.recvfrom(65535)

            pcap_file1.write(raw_data)
```

```python



        dest_addr, src_addr, eth_proto, data = ethernet_frame(raw_data)
        print('\n Ehternet Frame: ')
        print(TAB_1 + 'Destination: {}, Source: {}, Protocol: {}'.format(dest_addr, src_addr, eth_proto))


        # 8 for IPv4

        if eth_proto ==8:
            (version, header_length, ttl, proto, src, target, data) = ipv4_packet(data)
            print(TAB_1 + 'IPv4_PAcket: ')
            print(TAB_2 + 'Version: {}, header_length: {}, TTL(Time to Live): {}'.format(version, header_length, ttl))
            print(TAB_2 + 'Protocol: {}, Source: {}, Target: {}'.format(proto, src, target))



            # 1 for ICMP

            if proto ==1:
                (icmp_type, code, checksum, data) = ICMP_packet(data)
                print(TAB_1 + 'ICMP_PAcket: ')
                print(TAB_2 + 'Type: {}, Code: {}, Checksum: {}'.format(icmp_type, code, checksum))
                print(TAB_2 + 'Data: {}')
                print(format_multi_line(DATA_TAB_3, data))



            # 6 for TCP

            elif proto == 6:
                (src_port, dest_port, sequence, acknowledgement, flag_urg, flag_ack, flag_psh, flag_rst, flag_syn, flag_fin, data) = tcp_packet(data)
                print(TAB_1 + 'TCP Segment: ')
                print(TAB_2 + 'Source port: {}, Destination port: {}'.format(src_port, dest_port))
                print(TAB_2 + 'Sequence: {}, Acknowledgement: {}'.format(sequence,
acknowledgement))
                    print(TAB_2 + 'Flags')
                    print(TAB_3 + 'URG: {}, ACK: {}, PSH: {}, RST: {}, SYN: {}, FIN: {}'.format(flag_urg, flag_ack, flag_psh, flag_rst, flag_syn, flag_fin))
                    print(TAB_2 + 'Data: {}')
                    print(format_multi_line(DATA_TAB_3, data))



                # 17 for UDP

                elif proto == 17:
                    (src_port, dest_port, size, data) = udp_packet(data)
                    print(TAB_1 + 'UDP Segment: ')
                    print(TAB_2 + 'Source port: {}, Destination port: {}, Length: {}'.format(src_port, dest_port, size))
                    print(TAB_2 + 'Data: {}')
                    print(format_multi_line(DATA_TAB_3, data))


                # Other

                else:
                    print(TAB_1 + 'Data: {}')
                    print(format_multi_line(DATA_TAB_2, data))

            else:
                print('Data: {}')
                print(format_multi_line(DATA_TAB_1, data))




            # flush data

            pcap_file1.pcap_file.flush()

            if count>=rounds:
                break

        # Closing the file
        pcap_file1.close()
```

6.2 Code for reading a “.pcap” file using scrapy:

Implemented by Ashish Sahu

```

       file_path = input("Enter your file name or file path: "
          )

        f_read = rdpcap(file_path)

        size1 = len(f_read)

        for packet in range(size1):

            pkt = f_read[packet]
            print(" Packet Information: \n\n")

            hexdump(pkt)

            print("\n\nFull Info of Packet: \n")
            pkt.show()
```

6.3 Code for pcap class :

```

PCAP_GLOBAL_HEADER_FMT = '@ I H H i I I I '


# Global Header Values
PCAP_MAGICAL_NUMBER = 2712847316
PCAP_MJ_VERN_NUMBER = 2
PCAP_MI_VERN_NUMBER = 4
PCAP_LOCAL_CORECTIN = 0
PCAP_ACCUR_TIMSTAMP = 0
PCAP_MAX_LENGTH_CAP = 65535
PCAP_DATA_LINK_TYPE = 1


class Pcap:

    def __init__(self, filename, link_type=PCAP_DATA_LINK_TYPE
       ):
        self.pcap_file = open(filename, 'wb')
        self.pcap_file.write(struct.pack('@ I H H i I I I ',
           PCAP_MAGICAL_NUMBER, PCAP_MJ_VERN_NUMBER,
           PCAP_MI_VERN_NUMBER, PCAP_LOCAL_CORECTIN,
           PCAP_ACCUR_TIMSTAMP, PCAP_MAX_LENGTH_CAP, link_type
           ))
        print ("[+] Link Type : {}".format(link_type))

```

```python



    def write(self, data):
        ts_sec, ts_usec = map(int, str(time.time()).split('.'))
        length = len(data)
        self.pcap_file.write(struct.pack('@ I I I I', ts_sec, ts_usec, length, length))
        self.pcap_file.write(data)

    def close(self):
        self.pcap_file.close()
```

6.4 Code for Method used to show captured packets data at realtime:

Implemented by Abhijeet Singh

```python

# Unpack Ethernet Frame

def ethernet_frame(data):
    src_mac, dest_mac, proto = struct.unpack("! 6s 6s H", data[:14])
    return get_mac_addr(dest_mac), get_mac_addr(src_mac), socket.htons(proto), data[14:]


# Return properly formatted MAC Address

def get_mac_addr(bytes_addr):
    bytes_str = map('{:02x}'.format, bytes_addr)
    return ':'.join(bytes_str).upper()


# Unpacking Ipv4 Packets

def ipv4_packet(data):
    version_header_length = data[0]
    version = version_header_length >> 4
    header_length = (version_header_length & 15) * 4
    ttl, proto, src, target = struct.unpack('! 8x B B 2x 4s 4s', data[:20])
    return version, header_length, ttl, proto, ipv4(src), ipv4(target), data[header_length:]


# Properly formatted Ipv4 address

```

```
#127.0.0.1

def ipv4(addr):
    return '.'.join(map(str, addr))


# Unpack ICMP PAckets

def ICMP_packet(data):
    icmp_type, code, checksum = struct.unpack('! B B H', data
       [:4])
    return icmp_type, code, checksum, data[4:]

# Unpack TCP Packet

def tcp_packet(data):
    (src_port, dest_port, sequence, acknowledgement,
       offset_reserve_flags) = struct.unpack('! H H L L H',
       data[:14])
    offset = (offset_reserve_flags >> 12) * 4
    flag_urg = (offset_reserve_flags & 32) >> 5
    flag_ack = (offset_reserve_flags & 16) >> 4
    flag_psh = (offset_reserve_flags & 8) >> 3
    flag_rst = (offset_reserve_flags & 4) >> 2
    flag_syn = (offset_reserve_flags & 2) >> 1
    flag_fin = (offset_reserve_flags & 1)

    return src_port, dest_port, sequence, acknowledgement,
       flag_urg, flag_ack, flag_psh, flag_rst, flag_syn,
       flag_fin, data[offset :]


# Unpack UDP PAcket

def udp_packet(data):
    src_port, dest_port, size = struct.unpack('! H H 2x H',
       data[:8])
    return src_port, dest_port, size, data[8:]



# formats multi line data

def format_multi_line(prefix, string, size = 80):
    size -= len(prefix)
    if isinstance(string, bytes):
        string = ''.join(r'\x{:02x}'.format(byte) for byte in
           string)
        if size % 2:
            size -=1

        return '\n'.join([prefix + line for line in textwrap.wrap(
            string, size)])
```

6.4.1 Contribution of each member towards Project:

Member 1: Abhilasha Sharma

1. Writing documents :
    - Project Purpose
    - Basic definitions of relevant topics that includes
        - Introduction to sniffing
        - What is packet sniffing
        - Tools used for packet sniffing
        - What is wireshark and use of wireshark
    - Reference of websites
2. Creating PPT
    - Introduction to project
    - Basic purpose of the project
    - Brief introduction to sniffing
    - What are packet sniffers
    - Tools that are being used for packet sniffing
    - Wireshark and its usage
3. Contribution in creating the project:
    - Creating a pcap file when a new session for packet capture is created
    - Using socket to create connection
    - After capturing packet using Pcap class to store packet info
    - Showing real-time basic information of captured packet

Member 2: Ashish Sahu

1. Writing documents :
    - Wireshark Demonstration
    - Working of Wireshark
    - Choosing an interface to capture packet
    - Start capturing
    - How wireshark captures packets
    - How to analyze different packets(TCP, UDP,ICMP, DNS, etc) in wireshark
2. Creating PPT

- Demonstration of Wireshark
- Steps to capture packets in wireshark
- How to read pcap files in wireshark
- Analyze packets using a set of tools available in wireshark to gain information.

3. Contribution in creating the project:

   - Creating a Class “Pcap” that will take captured packet as input and store it in a “.pcap” file
   - Reading a “.pcap” from a given path
   - Printing all the information of packet captured

Member 3: Abhijeet Singh

1. Writing documents :

   - Description of packages that are used in project:
     - Scapy
     - Socket
     - Textwrap
     - Struct
   - Purpose of each package
   - Methodology used for creating project
   - References

2. Creating PPT

   - Packages used for working of the project
   - Brief description of each package any their functioning
   - Scrapy, textwrap, struct, socket, etc
   - Methodology used for creating the project

3. Contribution in creating the project:

   - Packages that are required for working of project
   - Writing methods for showing real-time description on each captured packet
   - Ex: icmp-segment, tcp-segment,udp-segment etc methods
   - Designing basic structure of packet sniffer
   - Adding comments at the required places

6.4.2 ***References:***

https://medium.com/@vworri/extracting-the-payload-from-a-pcap-file-using-python-d938d7622d71

https://subscription.packtpub.com/book/networking$_and_servers/9781784399771/8/ch08lvl1sec48/reading-and-writing-to-pcap-files$

https://cppsecrets.com/users/812115971219710810510797110971151014957494864103109971051084699111109/Python-Scapy-Reading-PCAP-file.php

https://youtu.be/WGJC5vT5YJo

https://www.bitforestinfo.com/blog/01/13/save-python-raw-tcpip-packet-into-pcap-files.html

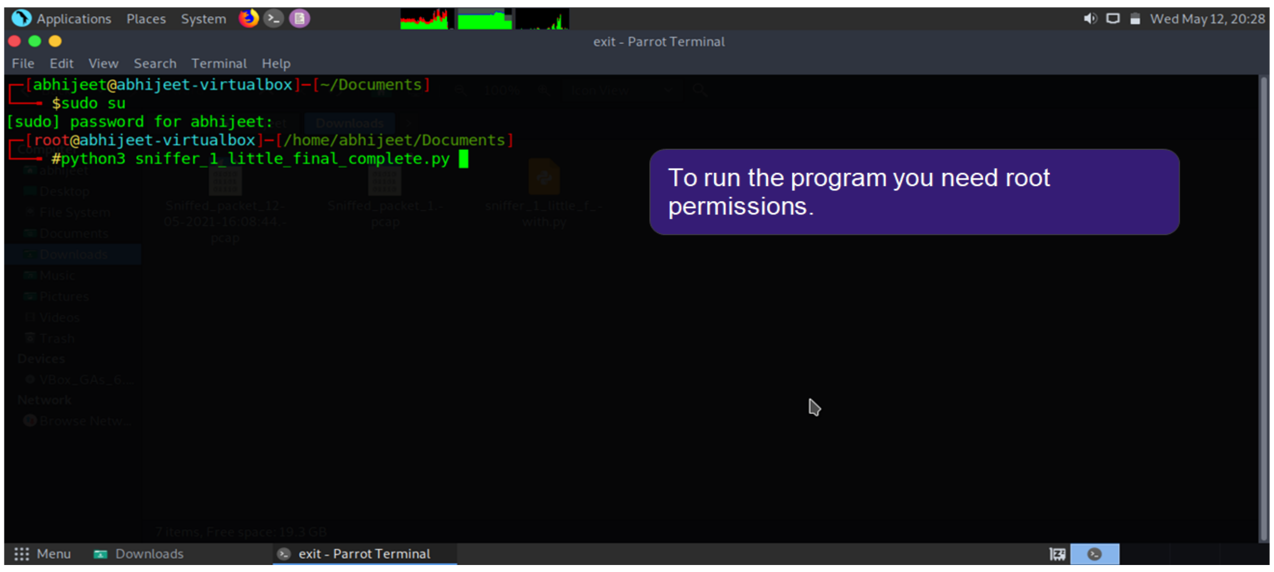
https://stackoverflow.com/questions/18256342/parsing-a-pcap-file-in-python

Chapter 7

Output of Project

7.0.1 Screenshot of running processes

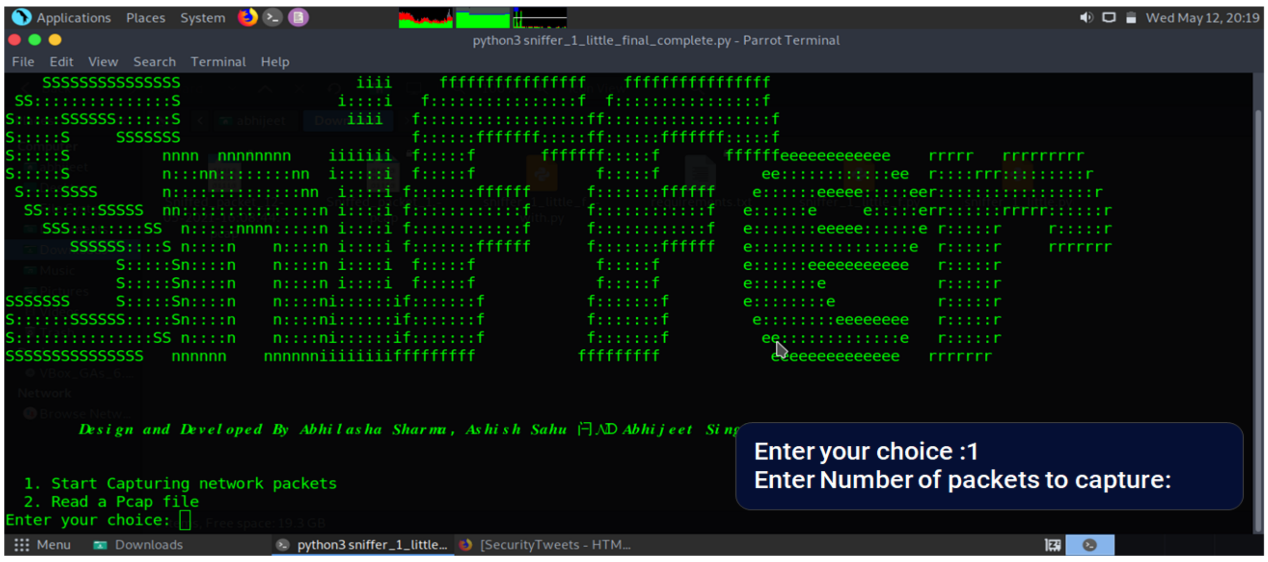
To run the project



```
┌─[abhijeet@abhijeet-virtualbox]─[~/Documents]
└──╼ $sudo su
[sudo] password for abhijeet:
┌─[root@abhijeet-virtualbox]─[/home/abhijeet/Documents]
└──╼ #python3 sniffer_1_little_final_complete.py
```

To run the program you need root permissions.

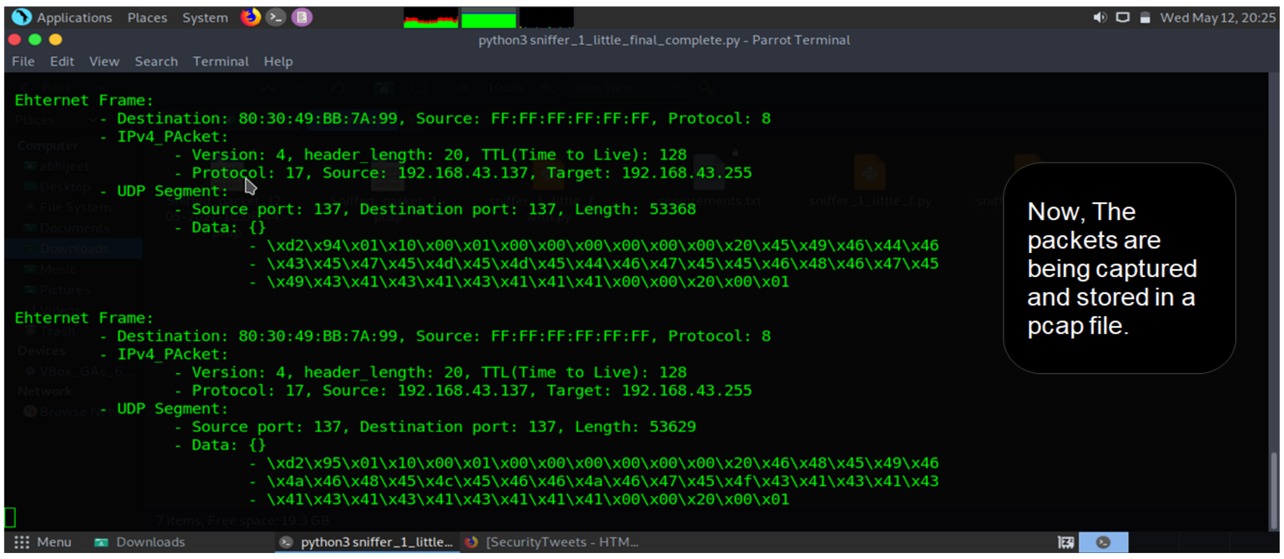
First Screen


Applications Places System
Wed May 12, 20:19
python3 sniffer_1_little_final_complete.py - Parrot Terminal
File Edit View Search Terminal Help
Menu Downloads python3 sniffer_1_little... [SecurityTweets - HTM...

```
    SSSSSSSSSSSSSSS                 iiii   ffffffffffffffff    ffffffffffffffff
  SS:::::::::::::::S               i::::i f::::::::::::::::f  f::::::::::::::::f
 S:::::SSSSSS::::::S                iiii f::::::::::::::::::ff::::::::::::::::::f
 S:::::S     SSSSSSS                     f::::::fffffff:::::ff::::::fffffff:::::f
 S:::::S            nnnn  nnnnnnnn    iiiiiii f:::::f       ffffff:::::f       ffffffeeeeeeeeeeee    rrrrr   rrrrrrrrr
 S:::::S            n:::nn::::::::nn  i:::::i f:::::f            f:::::f           ee::::::::::::ee  r::::rrr:::::::::r
  S::::SSSS         n::::::::::::::nn  i::::i f:::::::ffffff     f:::::::ffffff    e::::::eeeee:::::eer:::::::::::::::::r
   SS::::::SSSSS    nn:::::::::::::::n i::::i f::::::::::::f     f::::::::::::f   e::::::e     e:::::err::::::rrrrr::::::r
     SSS::::::::SS    n:::::nnnn:::::n i::::i f::::::::::::f     f::::::::::::f   e:::::::eeeee::::::e r:::::r     r:::::r
        SSSSSS::::S   n::::n    n::::n i::::i f:::::::ffffff     f:::::::ffffff   e:::::::::::::::::e  r:::::r     rrrrrrr
             S:::::S  n::::n    n::::n i::::i  f:::::f            f:::::f         e::::::eeeeeeeeeee   r:::::r
             S:::::S  n::::n    n::::n i::::i  f:::::f            f:::::f         e:::::::e            r:::::r
 SSSSSSS     S:::::S  n::::n    n::::ni::::::if:::::::f          f:::::::f        e::::::::e           r:::::r
 S::::::SSSSSS:::::S  n::::n    n::::ni::::::if:::::::f          f:::::::f         e::::::::eeeeeeee   r:::::r
 S:::::::::::::::SS   n::::n    n::::ni::::::if:::::::f          f:::::::f          ee:::::::::::::e   r:::::r
  SSSSSSSSSSSSSSS     nnnnnn    nnnnnniiiiiiiifffffffff          fffffffff            eeeeeeeeeeeeee   rrrrrrr


        Design and Developed By Abhilasha Sharma, Ashish Sahu 㐫ND Abhijeet Sing

  1. Start Capturing network packets
  2. Read a Pcap file
Enter your choice: 
```

Enter your choice :1
Enter Number of packets to capture:

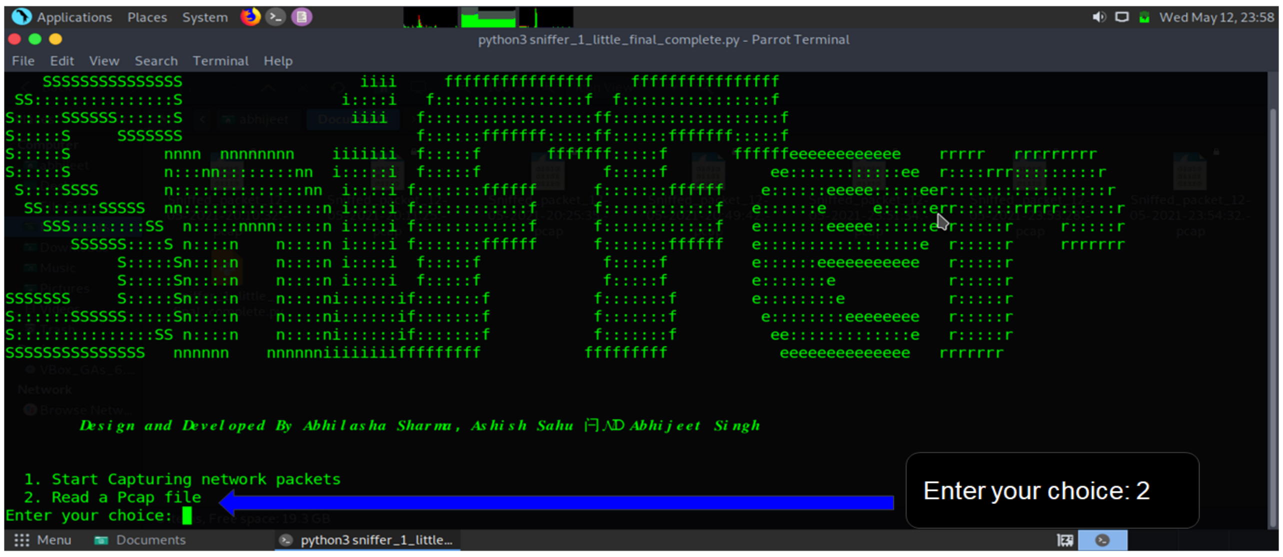
Capturing Packets


Applications Places System
Wed May 12, 20:25
python3 sniffer_1_little_final_complete.py - Parrot Terminal
File Edit View Search Terminal Help
Menu Downloads python3 sniffer_1_little... [SecurityTweets - HTM...

```
Ehternet Frame:
         - Destination: 80:30:49:BB:7A:99, Source: FF:FF:FF:FF:FF:FF, Protocol: 8
         - IPv4_PAcket:
                 - Version: 4, header_length: 20, TTL(Time to Live): 128
                 - Protocol: 17, Source: 192.168.43.137, Target: 192.168.43.255
         - UDP Segment:
                 - Source port: 137, Destination port: 137, Length: 53368
                 - Data: {}
                         - \xd2\x94\x01\x10\x00\x01\x00\x00\x00\x00\x00\x00\x20\x45\x49\x46\x44\x46
                         - \x43\x45\x47\x45\x4d\x45\x4d\x45\x44\x46\x47\x45\x45\x46\x48\x46\x47\x45
                         - \x49\x43\x41\x43\x41\x43\x41\x41\x41\x00\x00\x20\x00\x01

Ehternet Frame:
         - Destination: 80:30:49:BB:7A:99, Source: FF:FF:FF:FF:FF:FF, Protocol: 8
         - IPv4_PAcket:
                 - Version: 4, header_length: 20, TTL(Time to Live): 128
                 - Protocol: 17, Source: 192.168.43.137, Target: 192.168.43.255
         - UDP Segment:
                 - Source port: 137, Destination port: 137, Length: 53629
                 - Data: {}
                         - \xd2\x95\x01\x10\x00\x01\x00\x00\x00\x00\x00\x00\x20\x46\x48\x45\x49\x46
                         - \x4a\x46\x48\x45\x4c\x45\x46\x46\x4a\x46\x47\x45\x4f\x43\x41\x43\x41\x43
                         - \x41\x43\x41\x43\x41\x43\x41\x41\x41\x00\x00\x20\x00\x01
```

Now, The packets are being captured and stored in a pcap file.

To Read the Pcap file

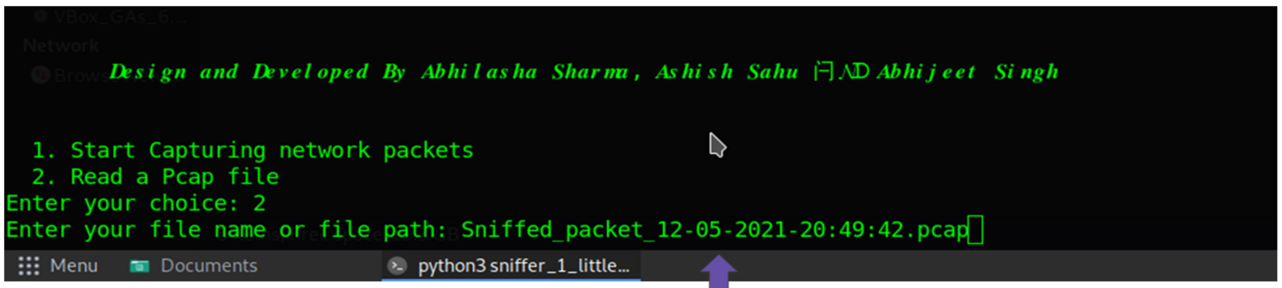


```
   SSSSSSSSSSSSSSS                 iiii     fffffffffffffff    fffffffffffffff
 SS:::::::::::::::S               i::::i   f::::::::::::::::f f::::::::::::::::f
S:::::SSSSSS::::::S                iiii   f::::::::::::::::::ff::::::::::::::::::f
S:::::S     SSSSSSS                       f::::::fffffff:::::ff::::::fffffff:::::f
S:::::S            nnnn  nnnnnnnn    iiiiiii f:::::f       ffffff:::::f       ffffffeeeeeeeeeeee    rrrrr   rrrrrrrrr
S:::::S            n:::nn::::::::nn  i:::::i f:::::f            f:::::f          ee::::::::::::ee  r::::rrr:::::::::r
 S::::SSSS         n::::::::::::::nn  i::::i f:::::::ffffff    f:::::::ffffff   e::::::eeeee:::::eer:::::::::::::::::r
  SS::::::SSSSS    nn:::::::::::::::n i::::i f::::::::::::f    f::::::::::::f  e::::::e     e:::::err::::::rrrrr::::::r
    SSS::::::::SS    n:::::nnnn:::::n i::::i f::::::::::::f    f::::::::::::f  e:::::::eeeee::::::e r:::::r     r:::::r
       SSSSSS::::S   n::::n    n::::n i::::i f:::::::ffffff    f:::::::ffffff  e:::::::::::::::::e  r:::::r     rrrrrrr
            S:::::S  n::::n    n::::n i::::i  f:::::f            f:::::f       e::::::eeeeeeeeeee   r:::::r
            S:::::S  n::::n    n::::n i::::i  f:::::f            f:::::f       e:::::::e            r:::::r
SSSSSSS     S:::::S  n::::n    n::::ni::::::if:::::::f          f:::::::f      e::::::::e           r:::::r
S::::::SSSSSS:::::S  n::::n    n::::ni::::::if:::::::f          f:::::::f       e::::::::eeeeeeee   r:::::r
S:::::::::::::::SS   n::::n    n::::ni::::::if:::::::f          f:::::::f        ee:::::::::::::e   r:::::r
 SSSSSSSSSSSSSSS     nnnnnn    nnnnnniiiiiiiifffffffff          fffffffff          eeeeeeeeeeeeee   rrrrrrr


        Design and Developed By Abhilasha Sharma, Ashish Sahu 闩ND Abhijeet Singh


  1. Start Capturing network packets
  2. Read a Pcap file
Enter your choice: 
```

Applications
Places
System
python3 sniffer_1_little_final_complete.py - Parrot Terminal
File
Edit
View
Search
Terminal
Help
Wed May 12, 23:58
Menu
Documents
python3 sniffer_1_little...

Enter your choice: 2

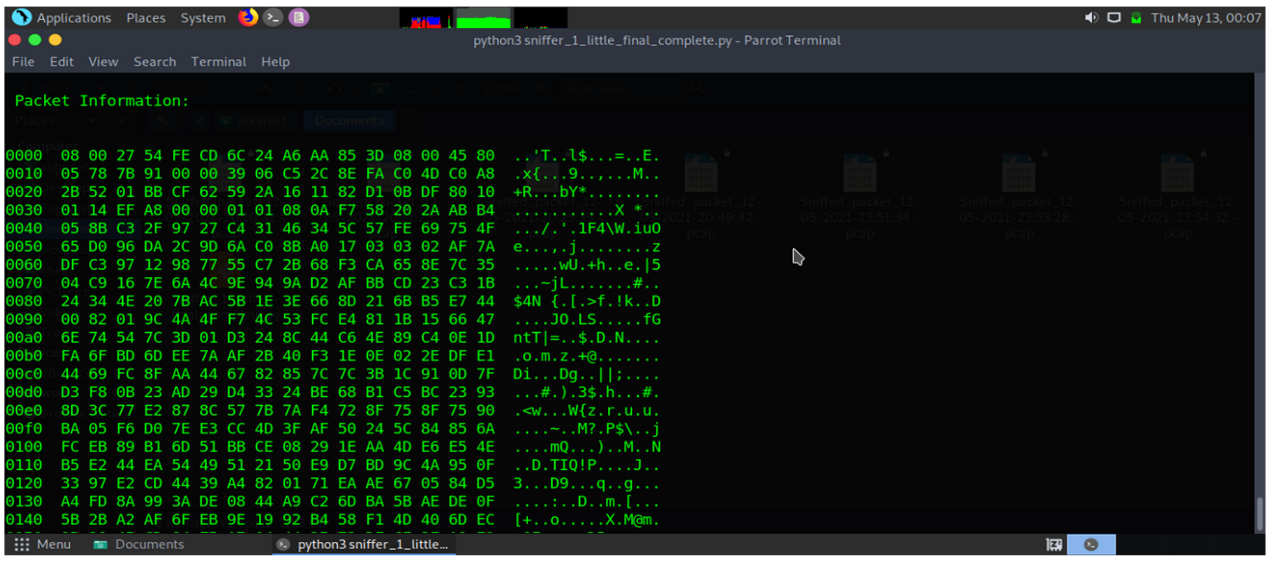
Give the path of Pcap file with its <filename.pcap>



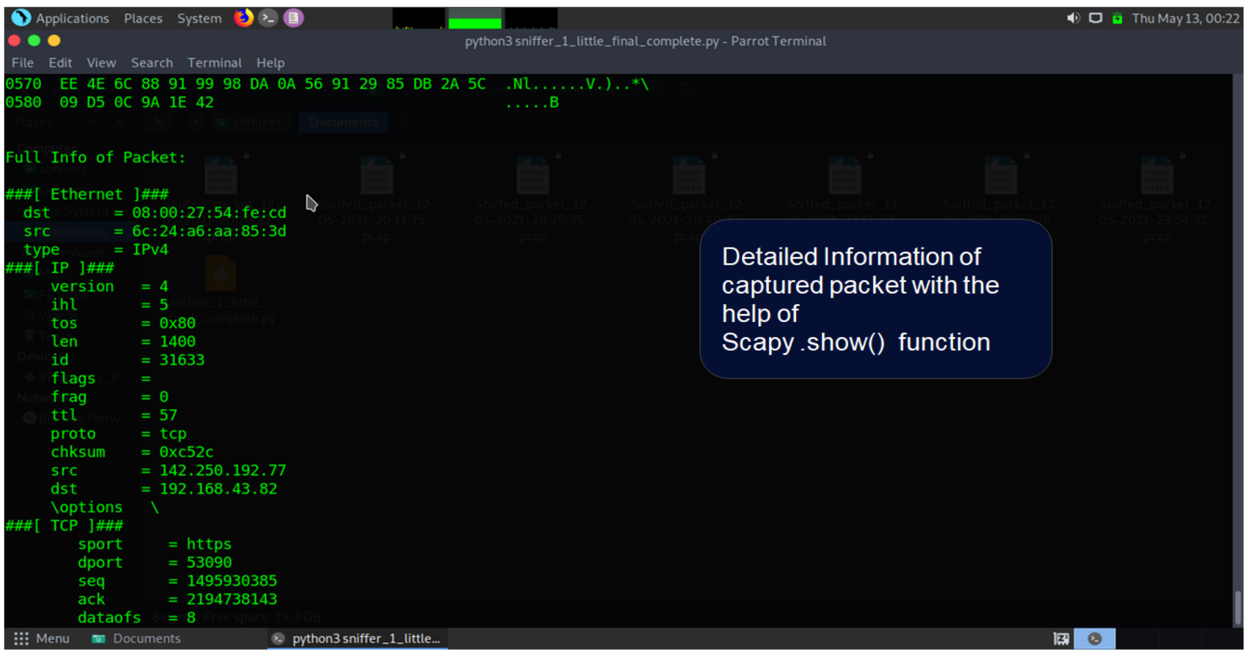
```
 Design and Developed By Abhilasha Sharma, Ashish Sahu 闩ND Abhijeet Singh

  1. Start Capturing network packets
  2. Read a Pcap file
Enter your choice: 2
Enter your file name or file path: Sniffed_packet_12-05-2021-20:49:42.pcap
```

Name of the file that we have to read

Data of a packet


Applications
Places
System
Thu May 13, 00:07
python3 sniffer_1_little_final_complete.py - Parrot Terminal
File
Edit
View
Search
Terminal
Help
Menu
Documents
python3 sniffer_1_little...

```
Packet Information:

0000  08 00 27 54 FE CD 6C 24 A6 AA 85 3D 08 00 45 80  ..'T..l$...=..E.
0010  05 78 7B 91 00 00 39 06 C5 2C 8E FA C0 4D C0 A8  .x{...9..,...M..
0020  2B 52 01 BB CF 62 59 2A 16 11 82 D1 0B DF 80 10  +R...bY*........
0030  01 14 EF A8 00 00 01 01 08 0A F7 58 20 2A AB B4  ...........X *..
0040  05 8B C3 2F 97 27 C4 31 46 34 5C 57 FE 69 75 4F  .../.'.1F4\W.iuO
0050  65 D0 96 DA 2C 9D 6A C0 8B A0 17 03 03 02 AF 7A  e...,.j........z
0060  DF C3 97 12 98 77 55 C7 2B 68 F3 CA 65 8E 7C 35  .....wU.+h..e.|5
0070  04 C9 16 7E 6A 4C 9E 94 9A D2 AF BB CD 23 C3 1B  ...~jL.......#..
0080  24 34 4E 20 7B AC 5B 1E 3E 66 8D 21 6B B5 E7 44  $4N {.[.>f.!k..D
0090  00 82 01 9C 4A 4F F7 4C 53 FC E4 81 1B 15 66 47  ....JO.LS.....fG
00a0  6E 74 54 7C 3D 01 D3 24 8C 44 C6 4E 89 C4 0E 1D  ntT|=..$.D.N....
00b0  FA 6F BD 6D EE 7A AF 2B 40 F3 1E 0E 02 2E DF E1  .o.m.z.+@.......
00c0  44 69 FC 8F AA 44 67 82 85 7C 7C 3B 1C 91 0D 7F  Di...Dg..||;....
00d0  D3 F8 0B 23 AD 29 D4 33 24 BE 68 B1 C5 BC 23 93  ...#.).3$.h...#.
00e0  8D 3C 77 E2 87 8C 57 7B 7A F4 72 8F 75 8F 75 90  .<w...W{z.r.u.u.
00f0  BA 05 F6 D0 7E E3 CC 4D 3F AF 50 24 5C 84 85 6A  ....~..M?.P$\..j
0100  FC EB 89 B1 6D 51 BB CE 08 29 1E AA 4D E6 E5 4E  ....mQ...)..M..N
0110  B5 E2 44 EA 54 49 51 21 50 E9 D7 BD 9C 4A 95 0F  ..D.TIQ!P....J..
0120  33 97 E2 CD 44 39 A4 82 01 71 EA AE 67 05 84 D5  3...D9...q..g...
0130  A4 FD 8A 99 3A DE 08 44 A9 C2 6D BA 5B AE DE 0F  ....:..D..m.[...
0140  5B 2B A2 AF 6F EB 9E 19 92 B4 58 F1 4D 40 6D EC  [+..o.....X.M@m.
```


Applications
Places
System
Thu May 13, 00:22
python3 sniffer_1_little_final_complete.py - Parrot Terminal
File
Edit
View
Search
Terminal
Help
Menu
Documents
python3 sniffer_1_little...

```
0570  EE 4E 6C 88 91 99 98 DA 0A 56 91 29 85 DB 2A 5C  .Nl......V.)..*\
0580  09 D5 0C 9A 1E 42                                .....B

Full Info of Packet:

###[ Ethernet ]###
  dst       = 08:00:27:54:fe:cd
  src       = 6c:24:a6:aa:85:3d
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x80
     len       = 1400
     id        = 31633
     flags     =
     frag      = 0
     ttl       = 57
     proto     = tcp
     chksum    = 0xc52c
     src       = 142.250.192.77
     dst       = 192.168.43.82
     \options   \
###[ TCP ]###
        sport     = https
        dport     = 53090
        seq       = 1495930385
        ack       = 2194738143
        dataofs   = 8
```

Detailed Information of captured packet with the help of
Scapy .show() function

Conclusion:

The most important purpose of this project is to develop a nifty tool(or a software) that have the ability to sniff over the network and capture data packets that is around it. Analyze those packets and gain some informative data from it to compromise the systems on its range. It can also be used to monitor a user's home network for activities that are being encapsulated in the process.There are many expensive tools out there for this purpose but it won't fits on an average students pocket money, so to help those people who really wants to know what is going on with their home network this project can be name "Network Sniffing With Wireshark Similar Tool". It would turn out to be a great help and guidance. From the current scenarios of this highly tempting and cunning world this tool come on the light that not just feasible for monitoring the network but can be used effective in the field of education as well.