Aim: Examine the use of packet sniffer tool: Wireshark

a) Download and install wireshark and capture different packets like icmp, tcp and http packets

b) Explore how the packets can be traced based on different filters

c) Capture packets of FTP and retrieve login ID and Password

Theory: -

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

Wireshark is used for:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color-coding and other features that let you dig deep into network traffic and inspect individual packets.

Features of Wireshark:

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.

Create various statistics.

Steps:

1. Open ubuntu terminal
2. Install wireshark
```
# apt-get install wireshark
```
3. To know the name of your Ethernet interface: (Mostly it is “etht0”)
```
#ifconfig
```
4. Start wireshark
```
#sudo wireshark
```
5. Once wireshark window opens, select the interface and click on start

a) Capturing Packets

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface.

For example, if you want to capture traffic on the wireless network, click your wireless interface.

You can configure advanced features by clicking Capture Options.

As soon as you click the interface‘s name, you‘ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.
Click the stop capture button near the top left corner of the window when you want to stop capturing traffic
Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.
Wireshark can record the capturing information in the file with extension .pcap (packet capture). This file can be again reopened for analysis in offline mode.
There is no need to remember filtering commands. Filters can be applied by putting predefined strings in Wireshark.

Commands:-

1. Capturing packets of a particular host

```
ip.addr = = 192.168.42.3
```

Sets a filter for any packet with 192.168.42.3, as either the source or destination.

2. To capture a conversation between specified hosts

```
ip.addr == 10.0.5.119 && ip.addr == 91.189.94.25
```

Sets a conversation filter between the two defined IP addresses.

b) Filtering Packets

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type ―dns and you‘ll see only DNS packets. When you start typing, Wireshark will help you auto complete your filter.

Commands:-

1. To filter packets for a specific protocol

```
http or dns
```

Sets a filter to display all http and dns requests.

2. To filter packets for specific port

```
tcp.port==4000
```

Sets a filter for any TCP packet with 4000 as a source or destination port.

3. Filter specific packets

```
tcp.flags.reset== 0
```

Displays all TCP resets.

4. Filter for http request packets

```
http.request
```

Displays all HTTP GET requests.

5. To filter traffic except given protocol packets

```
!(arp or icmp or dns)
```

Masks out arp, icmp, dns, or whatever other protocols may be background noise, allowing you to focus on the traffic of interest.

6. Capturing packets after applying multiple filters

```
not (tcp.port == 80) and not (tcp port == 25)
```

Get all packets which are not HTTP or UDP.

To stop capturing click on the “red square”

c) To capture packets of FTP server. (Login ID and Password)

What is FTP?
FTP stands for File Transfer Protocol. As the name suggest this network protocol allows you to transfer files or directories from one host to another over the network whether it is your LAN or Internet.

The package required to install FTP is known as VSFTPD (Very Secure File Transfer Protocol Daemon)

Steps:-
1. Get root access: $ sudo su root
2. Find your ip address: # ifconfig

<u>Installation of FTP server in Ubuntu</u>
Name of Packages required: VSFTPD, XINETD

1. # sudo apt-get install vsftpd
2. # sudo apt-get install xinetd
The above command will install and start the xinetd superserver on your system. The chances are that you already have xinetd installed on your system. In that case you can omit the above installation command.
In the next step we need to edit the FTP server's configuration file which is present in /etc/vsftpd.conf
3. # cd /etc
4. # ls
5. # gedit vsftpd.conf
Change the following line:
Anonymous_enable=NO To Anonymous_enable=YES
This will instruct the FTP server to allow connecting with an anonymous client.
6. Save and close the gedit file

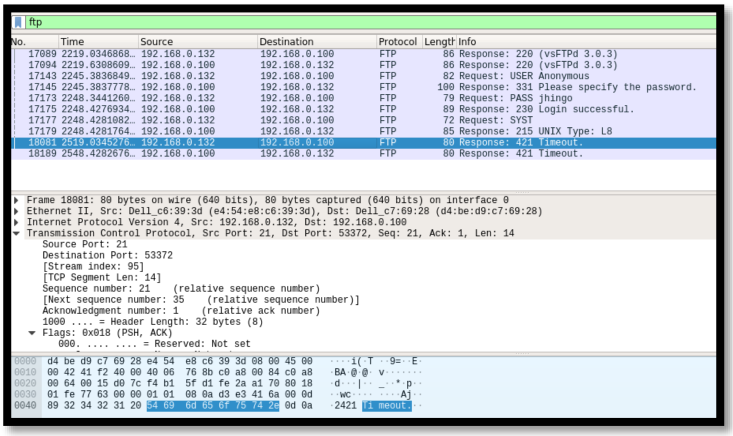
Now, that we are ready we can start the FTP server in the normal mode with:
7. # service xinetd restart
8. # service vsftpd restart OR # init.d/vsftpd restart

Start WIRESHARK. In the FILTER field put FTP. This will filter all FTP packets

Connectimg to a client present in other machine
$ ftp ip address of the FTP server
Name: anonymous
Please specify the password.
Password:
Login successful. (even if the login is not successful then also wireshark will capture the id and password)

```
ftp>
ftp> quit
Goodbye.
```

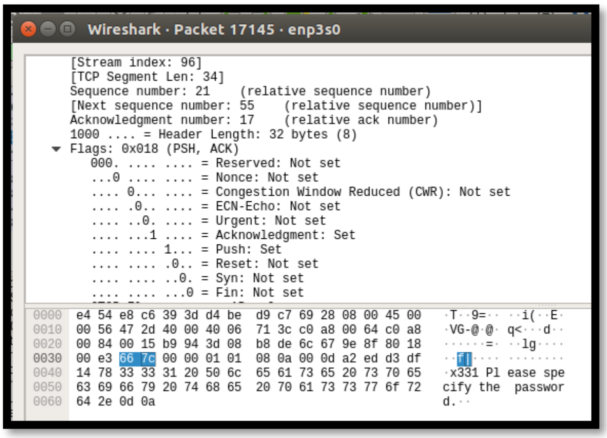
While the client is establishing a connection with the FTP server, the wireshark running in the background of the FTP server is able to capture all FTP packets. So, the Name and Password entered by the client is visible in plain text in Wireshark. Apart from that the source and destination address is also visible. If many clients are trying to connect with the server then source address, name and password are visible for all of them.



```
ftp
No.   Time            Source         Destination    Protocol Length Info
17089 2219.0346868… 192.168.0.132  192.168.0.100  FTP       86 Response: 220 (vsFTPd 3.0.3)
17094 2219.6308609… 192.168.0.100  192.168.0.132  FTP       86 Response: 220 (vsFTPd 3.0.3)
17143 2245.3836849… 192.168.0.132  192.168.0.100  FTP       82 Request: USER Anonymous
17145 2245.3837778… 192.168.0.100  192.168.0.132  FTP      100 Response: 331 Please specify the password.
17173 2248.3441260… 192.168.0.132  192.168.0.100  FTP       79 Request: PASS jhingo
17175 2248.4276934… 192.168.0.100  192.168.0.132  FTP       89 Response: 230 Login successful.
17177 2248.4281082… 192.168.0.132  192.168.0.100  FTP       72 Request: SYST
17179 2248.4281764… 192.168.0.100  192.168.0.132  FTP       85 Response: 215 UNIX Type: L8
18081 2519.0345276… 192.168.0.132  192.168.0.100  FTP       80 Response: 421 Timeout.
18189 2548.4282676… 192.168.0.100  192.168.0.132  FTP       80 Response: 421 Timeout.

▸ Frame 18081: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
▸ Ethernet II, Src: Dell_c6:39:3d (e4:54:e8:c6:39:3d), Dst: Dell_c7:69:28 (d4:be:d9:c7:69:28)
▸ Internet Protocol Version 4, Src: 192.168.0.132, Dst: 192.168.0.100
▾ Transmission Control Protocol, Src Port: 21, Dst Port: 53372, Seq: 21, Ack: 1, Len: 14
    Source Port: 21
    Destination Port: 53372
    [Stream index: 95]
    [TCP Segment Len: 14]
    Sequence number: 21    (relative sequence number)
    [Next sequence number: 35    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▾ Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set

0000  d4 be d9 c7 69 28 e4 54  e8 c6 39 3d 08 00 45 00   ····i(·T ··9=··E·
0010  00 42 41 f2 40 00 40 06  76 8b c0 a8 00 84 c0 a8   ·BA·@·@· v·······
0020  00 64 00 15 d0 7c f4 b1  5f d1 fe 2a a1 70 80 18   ·d···|·· _··*·p··
0030  01 fe 77 63 00 00 01 01  08 0a d3 e3 41 6a 00 0d   ··wc···· ····Aj··
0040  89 32 34 32 31 20 54 69  6d 65 6f 75 74 2e 0d 0a   ·2421 Ti meout.··
```



Wireshark · Packet 17145 · enp3s0

```
    [Stream index: 96]
    [TCP Segment Len: 34]
    Sequence number: 21    (relative sequence number)
    [Next sequence number: 55    (relative sequence number)]
    Acknowledgment number: 17    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▾ Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set

0000  e4 54 e8 c6 39 3d d4 be  d9 c7 69 28 08 00 45 00   ·T··9=·· ··i(··E·
0010  00 56 47 2d 40 00 40 06  71 3c c0 a8 00 64 c0 a8   ·VG-@·@· q<···d··
0020  00 84 00 15 b9 94 3d 08  b8 de 6c 67 9e 8f 80 18   ······=· ··lg····
0030  00 e3 66 7c 00 00 01 01  08 0a 00 0d a2 ed d3 df   ··f|···· ········
0040  14 78 33 33 31 20 50 6c  65 61 73 65 20 73 70 65   ·x331 Pl ease spe
0050  63 69 66 79 20 74 68 65  20 70 61 73 73 77 6f 72   cify the  passwor
0060  64 2e 0d 0a                                        d.··
```