

Wireshark Implementation

What is Wireshark?

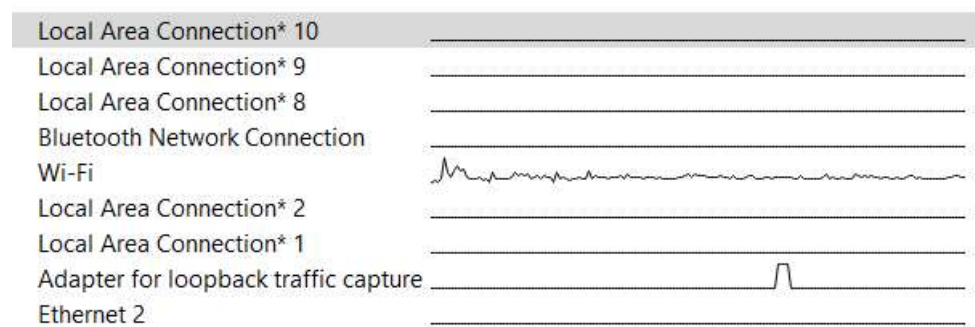
Wireshark is a traffic monitoring and packet analysis tool. It is used for network troubleshooting and detect suspicion activities. It is free and open-source software, thus, is used by many for educational and research purposes.

Various Uses of Wireshark

1. Capture traffic on any network
2. Analyse Traffic
3. Analyse Packets
4. Password Sniffing

1. Capturing Traffic on Any Network

The main screen shows different networks connected to the computer system. Network showing graphs indicate there is a traffic on this network.



For our purpose, let's examine the **VIT Wi-fi network**.

As soon as we click on Wi-Fi, capturing of traffic starts.

1520	14.402462	HewlettP_88:b5:68	Broadcast	ARP	60 Who has 172.16.91.153? Tell 172.16.88.1
1521	14.402462	IntelCor_cb:d3:c9	Broadcast	ARP	60 Who has 169.254.255.255? Tell 172.16.91.44
1522	14.402462	IntelCor_7c:b3:ca	Broadcast	ARP	60 Who has 169.254.255.255? Tell 172.16.93.233
1523	14.511368	51.79.234.100	172.16.93.244	TLSv1.2	93 Application Data
1524	14.511568	172.16.93.244	51.79.234.100	TCP	54 63764 → 443 [FIN, ACK] Seq=1 Ack=40 Win=510 Len=0
1525	14.511705	172.16.91.44	172.16.95.255	UDP	305 54915 → 54915 Len=263
1526	14.512607	51.79.234.100	172.16.93.244	TLSv1.2	78 Application Data
1527	14.512607	51.79.234.100	172.16.93.244	TCP	60 443 → 63764 [FIN, ACK] Seq=64 Ack=1 Win=83 Len=0
1528	14.512716	172.16.93.244	51.79.234.100	TCP	54 63764 → 443 [RST, ACK] Seq=2 Ack=64 Win=0 Len=0
1529	14.525907	198.22.162.81	172.16.93.244	TCP	60 443 → 63752 [ACK] Seq=745 Ack=3224 Win=91 Len=0
1530	14.533684	198.22.162.81	172.16.93.244	TLSv1.2	302 Application Data

Screenshot of traffic at a particular instance. (It is in encrypted form)

There are different colouring schemes for different cases. These colouring rules are set by the wireshark software, and can be customized according to our needs.

Name	Filter
<input checked="" type="checkbox"/> Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack
<input checked="" type="checkbox"/> HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/> Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/> OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/> ICMP errors	icmp.type eq 3 icmp.type eq 4 icmp.type eq 5 icmp.type eq 11 icmpv6.type eq 1 icmpv6.type eq 2 icmpv6.type eq 3 icmpv6.type eq 4
<input checked="" type="checkbox"/> ARP	arp
<input checked="" type="checkbox"/> ICMP	icmp icmpv6
<input checked="" type="checkbox"/> TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/> SCTP ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/> TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !ipm && !ospf) (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp carp))
<input checked="" type="checkbox"/> Checksum Errors	eth.fcs.status=="Bad" ip.checksum.status=="Bad" tcp.checksum.status=="Bad" udp.checksum.status=="Bad" sctp.checksum.status=="Bad" mstp
<input checked="" type="checkbox"/> SMB	smb nbss nbns netbios
<input checked="" type="checkbox"/> HTTP	http tcp.port == 80 http2
<input checked="" type="checkbox"/> DCERPC	dcerpc
<input checked="" type="checkbox"/> Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ismp
<input checked="" type="checkbox"/> TCP SYN/FIN	tcp.flags & 0x02 tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP	tcp
<input checked="" type="checkbox"/> UDP	udp
<input checked="" type="checkbox"/> Broadcast	eth[0] & 1
<input checked="" type="checkbox"/> System Event	systemd_journal sysdig

Toolbar → View → Coloring rules

2. Analyse Traffic

After stopping the capturing of data, we will have a huge amount of traffic data. This can be analysed to obtain information regarding network traffic, data sent and received, ip addresses involved, etc.

There are various types of protocols in our traffic data. We can filter it according to our needs.

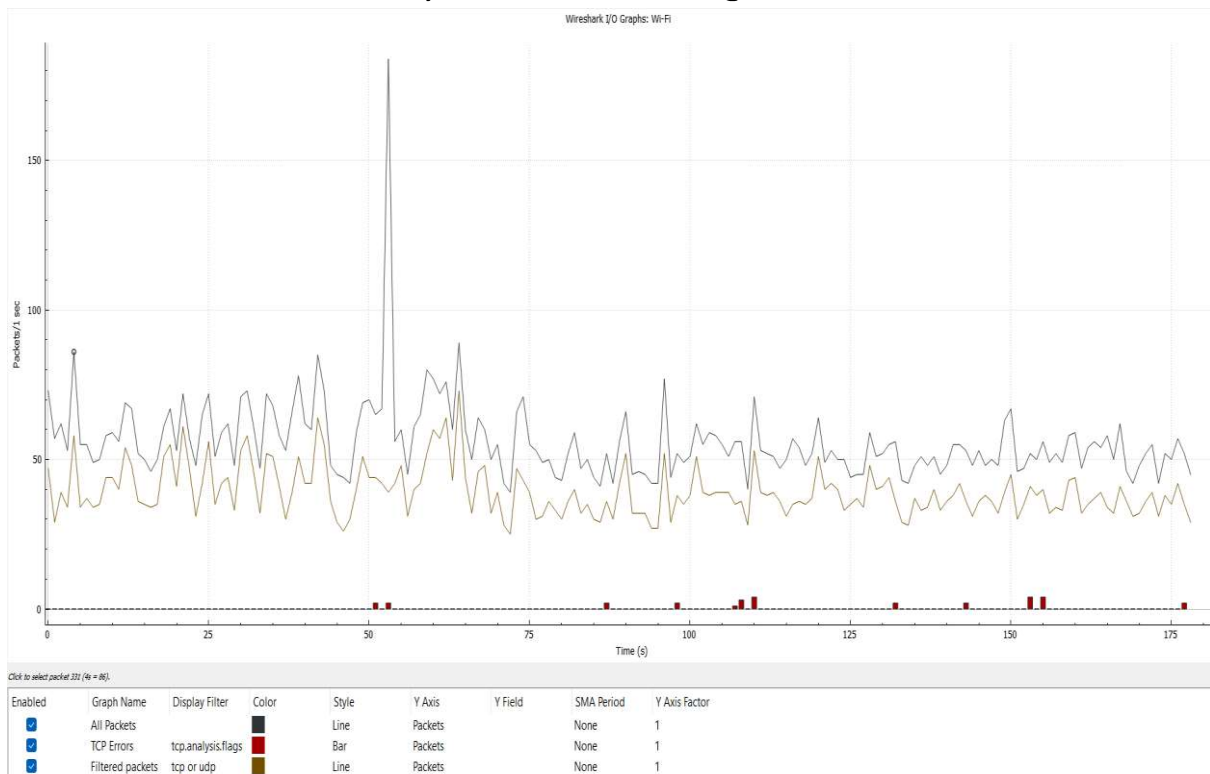
tcp						
No.	Time	Source	Destination	Protocol	Length	Info
37	0.469272	13.107.4.52	172.16.93.244	TCP	60	80 → 64532 [FIN, ACK] Seq=540 Ack=112 Win=4194304 Len=0
38	0.469394	172.16.93.244	13.107.4.52	TCP	54	64532 → 80 [ACK] Seq=112 Ack=541 Win=131840 Len=0
39	0.469538	172.16.93.244	13.107.4.52	TCP	54	64532 → 80 [FIN, ACK] Seq=112 Ack=541 Win=131840 Len=0
41	0.494585	13.107.4.52	172.16.93.244	TCP	60	80 → 64532 [ACK] Seq=541 Ack=113 Win=4194304 Len=0
163	2.547473	172.16.93.244	20.198.162.76	TLSv1.2	98	Application Data
167	2.628771	20.198.162.76	172.16.93.244	TLSv1.2	229	Application Data
172	2.672087	172.16.93.244	20.198.162.76	TCP	54	61039 → 443 [ACK] Seq=45 Ack=176 Win=258 Len=0
431	6.886087	13.227.166.100	172.16.93.244	TLSv1.2	93	Application Data
432	6.886087	13.227.166.100	172.16.93.244	TLSv1.2	78	Application Data
433	6.886087	13.227.166.100	172.16.93.244	TCP	60	443 → 64424 [FIN, ACK] Seq=64 Ack=1 Win=131 Len=0
434	6.886200	172.16.93.244	13.227.166.100	TCP	54	64424 → 443 [ACK] Seq=1 Ack=65 Win=257 Len=0
502	8.335310	172.16.93.244	104.85.157.227	TCP	55	64509 → 443 [ACK] Seq=1 Ack=1 Win=509 Len=1 [TCP segment of a reassembled PDU]

For example, if we type 'tcp' in display filter, we obtain a list of only those data packets which has TCP protocol. We can also use 'tcp contains youtube'.

tcp or udp						
No.	Time	Source	Destination	Protocol	Length	Info
29	0.426212	172.16.94.113	172.16.95.255	UDP	62	2007 → 2007 Len=20
32	0.441568	13.107.4.52	172.16.93.244	TCP	66	80 → 64532 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
33	0.441699	172.16.93.244	13.107.4.52	TCP	54	64532 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
34	0.441910	172.16.93.244	13.107.4.52	HTTP	165	GET /connecttest.txt HTTP/1.1
35	0.467244	13.107.4.52	172.16.93.244	TCP	60	80 → 64532 [ACK] Seq=1 Ack=112 Win=4194304 Len=0
36	0.469272	13.107.4.52	172.16.93.244	HTTP	593	HTTP/1.1 200 OK (text/plain)
37	0.469272	13.107.4.52	172.16.93.244	TCP	60	80 → 64532 [FIN, ACK] Seq=540 Ack=112 Win=4194304 Len=0
38	0.469394	172.16.93.244	13.107.4.52	TCP	54	64532 → 80 [ACK] Seq=112 Ack=541 Win=131840 Len=0
39	0.469538	172.16.93.244	13.107.4.52	TCP	54	64532 → 80 [FIN, ACK] Seq=112 Ack=541 Win=131840 Len=0
40	0.492297	172.16.94.76	172.16.95.255	UDP	305	54915 → 54915 Len=263
41	0.494585	13.107.4.52	172.16.93.244	TCP	60	80 → 64532 [ACK] Seq=541 Ack=113 Win=4194304 Len=0
42	0.506081	172.16.92.193	172.16.95.255	UDP	305	54915 → 54915 Len=263

Filtering 'tcp or udp' protocols. For 'https', we use 'tls'.

We can also obtain analysis of traffic using wireshark.



Toolbar → Statistics → I/O Graphs

We can also obtain summary of conversations between endpoints.

Ethernet · 119 IPv4 · 114 IPv6 · 131 TCP · 18 UDP · 408											
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
0.0.0.0	255.255.255.255	38	13 k	38	13 k	0	0	0.024572	165.2740	645	0
13.227.166.100	172.16.93.244	9	568	5	351	4	217	6.886087	57.8600	48	30
172.16.88.54	239.255.255.250	18	3742	18	3742	0	0	0.895904	141.9653	210	0
172.16.88.54	172.16.95.255	3	258	3	258	0	0	0.39.812043	107.5178	19	0
172.16.88.54	224.0.0.251	1	87	1	87	0	0	0.117.631337	0.0000	—	0
172.16.88.165	239.255.255.250	4	864	4	864	0	0	0.102.898831	3.0287	2282	0
172.16.88.185	224.0.0.251	7	2338	7	2338	0	0	0.92.611939	26.1419	715	0
172.16.88.195	172.16.95.255	9	828	9	828	0	0	0.22.472409	121.1631	54	0
172.16.89.3	239.255.255.250	4	864	4	864	0	0	0.101.524946	3.0195	2289	0
172.16.89.29	239.255.255.250	8	1724	8	1724	0	0	0.73.683071	3.0388	4538	0
172.16.89.40	239.255.255.250	7	1505	7	1505	0	0	0.55.897462	122.0307	98	0
172.16.89.210	172.16.95.255	178	54 k	178	54 k	0	0	0.0.970427	176.9967	2453	0
172.16.89.210	239.255.255.250	4	860	4	860	0	0	0.115.990697	3.0214	2277	0
172.16.89.237	239.255.255.250	4	864	4	864	0	0	0.6.066878	3.0243	2285	0
172.16.90.2	172.16.95.255	7	574	7	574	0	0	0.12.692565	161.8442	28	0
172.16.90.37	224.0.0.251	8	952	8	952	0	0	0.19.068507	140.0232	54	0
172.16.90.148	239.255.255.250	16	3416	16	3416	0	0	0.39.372223	123.0468	222	0
172.16.90.163	239.255.255.250	4	864	4	864	0	0	0.90.557432	3.0346	2277	0
172.16.90.181	239.255.255.250	16	3456	16	3456	0	0	0.0.255966	128.4495	215	0
172.16.90.186	239.255.255.250	8	1728	8	1728	0	0	0.50.928176	123.0639	112	0
172.16.90.242	172.16.95.255	33	3036	33	3036	0	0	0.17.795938	143.2692	169	0
172.16.91.7	172.16.95.255	3	276	3	276	0	0	0.142.392067	1.5115	1460	0
172.16.91.38	239.255.255.250	4	860	4	860	0	0	0.116.879231	2.9917	2299	0
172.16.91.89	239.255.255.250	4	864	4	864	0	0	0.59.537401	3.0457	2269	0
172.16.91.241	239.255.255.250	8	1728	8	1728	0	0	0.50.751734	123.0124	112	0
172.16.91.251	239.255.255.250	8	1720	8	1720	0	0	0.29.385353	123.0311	111	0
172.16.92.5	172.16.95.255	120	11 k	120	11 k	0	0	0.4.1983257	173.8613	507	0

☐ Name resolution ☐ Limit to display filter ☐ Absolute start time

Conversation Types ▾

Toolbar → Statistics → Conversations

3. Analyse Packets

There are many data packets being sent and received across network at any instance of time.

Upon selecting a particular packet, we obtain greater details from the Status pane below.

For example, upon selecting the given data packet

33 0.441699 172.16.93.244 13.107.4.52 TCP 54 64532 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0

few of the details are already displayed in the list such as serial number, **relative** time of data packet being transferred, source address, destination address, protocol, length of data packet in bytes and other information.

In the Status pane below,

```
> Frame 33: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{62301BF6-BE9D-44C8-AB16-8FD4F71014AE}, id 0
> Ethernet II, Src: IntelCor_c8:08:fc (94:e2:3c:c8:08:fc), Dst: HewlettP_88:b5:68 (78:ac:c0:88:b5:68)
> Internet Protocol Version 4, Src: 172.16.93.244, Dst: 13.107.4.52
v Transmission Control Protocol, Src Port: 64532, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 64532
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1 (relative sequence number)
    Sequence Number (raw): 140290109
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 1 (relative ack number)
```

We can see greater details such as source port number, destination port number, etc.

The data info is shown in the raw hexadecimal form.

```
0000 94 e2 3c c8 08 fc 78 ac c0 88 b5 68 08 00 45 00  --<...x...h..E..
0010 01 9d eb 53 00 00 40 11 7f e6 ac 10 58 01 ac 10  --S..@...X...
0020 5d f4 00 35 eb 9c 01 89 ba bc 88 53 81 80 00 01  ]..5....S....
0030 00 05 00 0d 00 00 03 77 77 77 0f 6d 73 66 74 63  ....w ww.msftc
0040 6f 6e 6e 65 63 74 74 65 73 74 03 63 6f 6d 00 00  onnectte st.com..
0050 01 00 01 c0 0c 00 05 00 01 00 00 04 b6 00 1d 08  ....
0060 6e 63 73 69 2d 67 65 6f 0e 74 72 61 66 66 69 63  ncsi-geo .traffic
0070 6d 61 6e 61 67 65 72 03 6e 65 74 00 c0 35 00 05  manager. net..5..
0080 00 01 00 00 06 2f 00 10 06 76 34 6e 63 73 69 06  ..../. v4ncsi-
0090 6d 73 65 64 67 65 c0 4d c0 5e 00 05 00 01 00 00  msedge.M ^.....
00a0 00 1a 00 19 04 6e 63 73 69 08 34 2d 63 2d 30 30  ....ncs i.4-c-00
00b0 30 33 08 63 2d 6d 73 65 64 67 65 c0 4d c0 7a 00  03.c-mse dge.M.z.
00c0 05 00 01 00 00 00 2a 00 02 c0 7f c0 7f 00 01 00  ....*. ....
00d0 01 00 00 00 08 00 04 0d 6b 04 34 00 00 02 00 01  .... k.4.....
```

We can also filter out data packets being sent or received from a particular ip address.

ip.addr ==172.16.93.244				
No.	Time	Source	Destination	Protocol
25	0.412596	172.16.93.244	172.16.88.1	DNS
26	0.414883	172.16.88.1	172.16.93.244	DNS
27	0.416103	172.16.93.244	13.107.4.52	TCP
32	0.441568	13.107.4.52	172.16.93.244	TCP
33	0.441699	172.16.93.244	13.107.4.52	TCP
34	0.441910	172.16.93.244	13.107.4.52	HTTP
35	0.467244	13.107.4.52	172.16.93.244	TCP
36	0.469272	13.107.4.52	172.16.93.244	HTTP
37	0.469272	13.107.4.52	172.16.93.244	TCP
38	0.469394	172.16.93.244	13.107.4.52	TCP
39	0.469538	172.16.93.244	13.107.4.52	TCP
41	0.494585	13.107.4.52	172.16.93.244	TCP

In this, only those packets are displayed which has its source or destination as ip.addr==172.16.93.244

If we want only those packets whose source is this ip address, then ip.src==172.16.93.244

And for destination, ip.dst==172.16.93.244

We can filter out using port addresses too.

tcp.port==463

4. Password Sniffing

The info of the packets is encrypted on most of the sites so we cannot decipher it.

To demonstrate password sniffing, let us use an unsecured login test site, exclusively used for such demonstration purposes.

Site : [login page \(vulnweb.com\)](http://vulnweb.com)

While demonstrating password sniffing, wireshark must continue capturing traffic on the network.

Username :	<input type="text" value="test"/>
Password :	<input type="password" value="...."/>
<input type="button" value="login"/>	

You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

This is for demonstration purpose only. The ID used is 'test' and password is 'test'.

Using the command `http.request.method=="POST"` in display filter, we get

http.request.method=="POST"						
No.	Time	Source	Destination	Protocol	Length	Info
1253	23.376050	172.16.93.244	44.228.249.3	HTTP	708	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
1921	36.377835	172.16.93.244	44.228.249.3	HTTP	831	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)

On clicking the userinfo.php encoded, we can see the ID and Password in the screen below.

```
> Frame 1253: 708 bytes on wire (5664 bits), 708 bytes captured (5664 bits) on interface \Device\NPF_{62301BF6-BE9D-44C8-AB16-8FD4F71014AE}, id 0
> Ethernet II, Src: IntelCor_c8:08:fc (94:e2:3c:c8:08:fc), Dst: HewlettP_88:b5:68 (78:ac:c0:88:b5:68)
> Internet Protocol Version 4, Src: 172.16.93.244, Dst: 44.228.249.3
> Transmission Control Protocol, Src Port: 57291, Dst Port: 80, Seq: 453, Ack: 2749, Len: 654
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded

▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "uname" = "test"
  > Form item: "pass" = "test"
```

Thus, Wireshark enabled by these use cases, is used by cyber security officials, network security engineers and system admins across various places to check for sniffing activities, network issues, performance issues or any troubleshooting with respect to network, and as an analysis tool.

-----Thank you-----