

# Mawlana Bhashani Science and Technology University



## Lab-Report

**Lab Report No : 08**

**Lab Report Name : Installing Wireshark in Linux Operating System**

**Course Title : Computer Networks Lab**

**Submitted by**

Name: Moniruzzaman & Mst.Zakia Sultana

ID: IT-18007 & IT-18027

3<sup>rd</sup> year 2<sup>nd</sup> semester

Session: 2017-2018

Dept. of ICT

MBSTU.

**Submitted To**

Nazrul Islam

Assistant Professor

Dept. of ICT

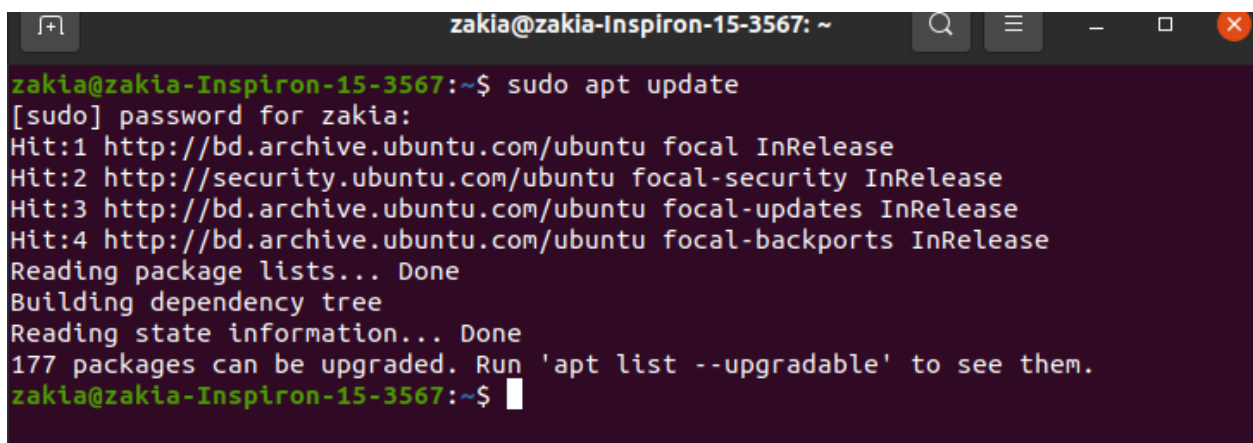
MBSTU.

## **INSTALLING WIRESHARK:**

Wireshark is a network packet analyzer. It captures every packet getting in or out of a network interface and shows them in a nicely formatted text. It is used by Network Engineers all over the world. How to install Wireshark is given below step by step: First update the APT package repository cache with the following command:

```
$ sudo apt update
```

The APT package repository cache should be updated.

A terminal window titled 'zakia@zakia-Inspiron-15-3567: ~' with a dark purple background. The terminal shows the execution of 'sudo apt update'. It prompts for the password for 'zakia', then lists four repository hits from 'bd.archive.ubuntu.com' for 'focal', 'focal-security', 'focal-updates', and 'focal-backports', all in 'InRelease' state. It then reports 'Reading package lists... Done', 'Building dependency tree', and 'Reading state information... Done'. Finally, it states '177 packages can be upgraded. Run 'apt list --upgradable' to see them.' and returns to the shell prompt.

```
zakia@zakia-Inspiron-15-3567: ~$ sudo apt update
[sudo] password for zakia:
Hit:1 http://bd.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://bd.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://bd.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
177 packages can be upgraded. Run 'apt list --upgradable' to see them.
zakia@zakia-Inspiron-15-3567:~$
```

Now, Run the following command to install Wireshark on your Ubuntu machine:

```
$ sudo apt get install wireshark
```

Wireshark should be installed.

Run the following command to add your user to the Wireshark group:

```
$ sudo usermod -aG wireshark $(whoami)
```

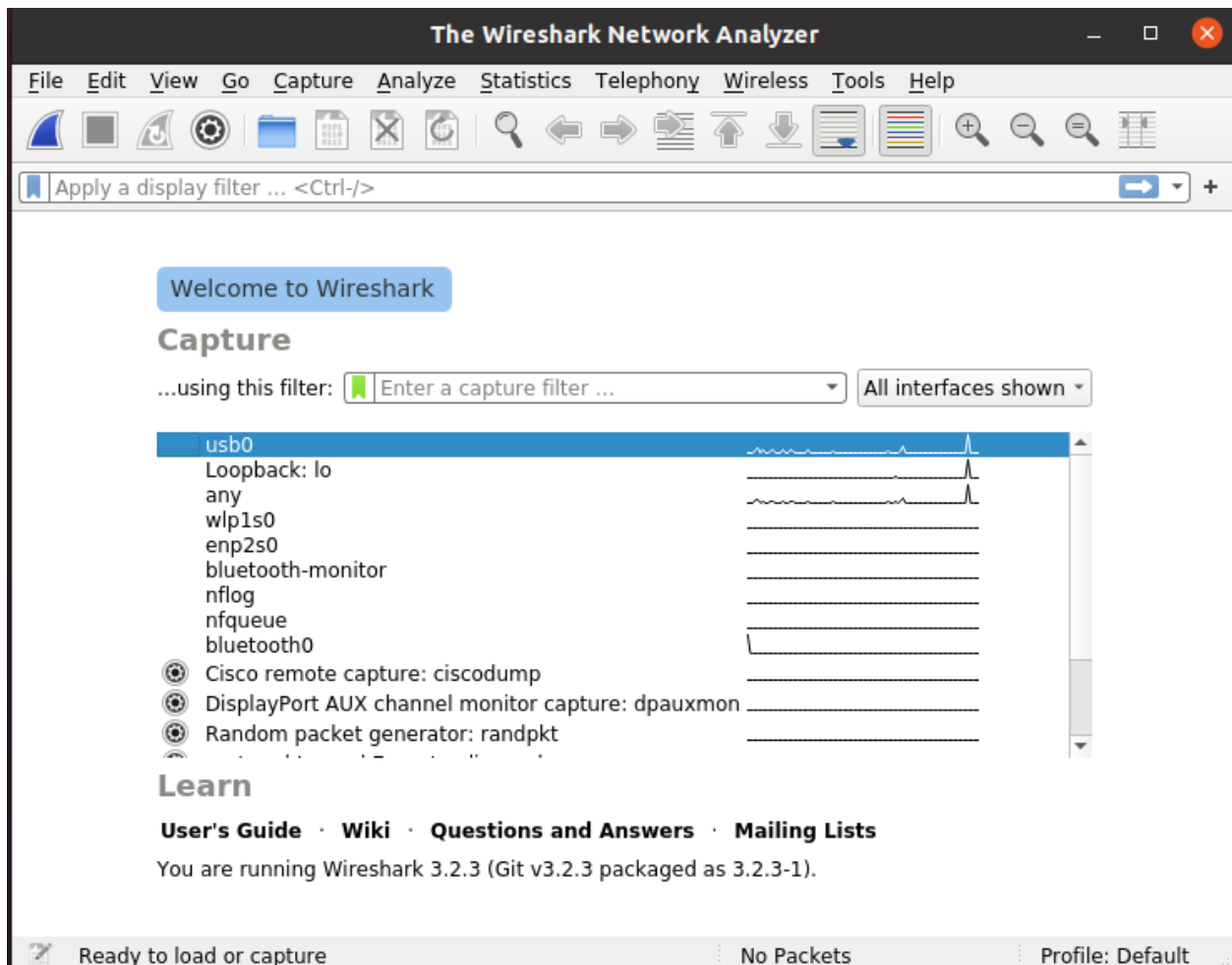
Now reboot your computer with the following command:

```
$ sudo reboot
```

Now run Wireshark using the following command:

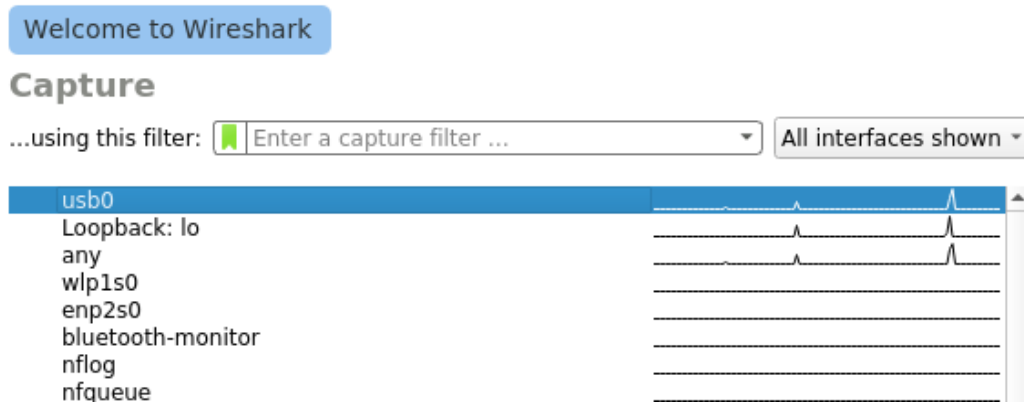
```
$ sudo wireshark
```

```
zakia@zakia-Inspiron-15-3567: ~  
zakia@zakia-Inspiron-15-3567:~$ sudo wireshark  
[sudo] password for zakia:  
OSStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

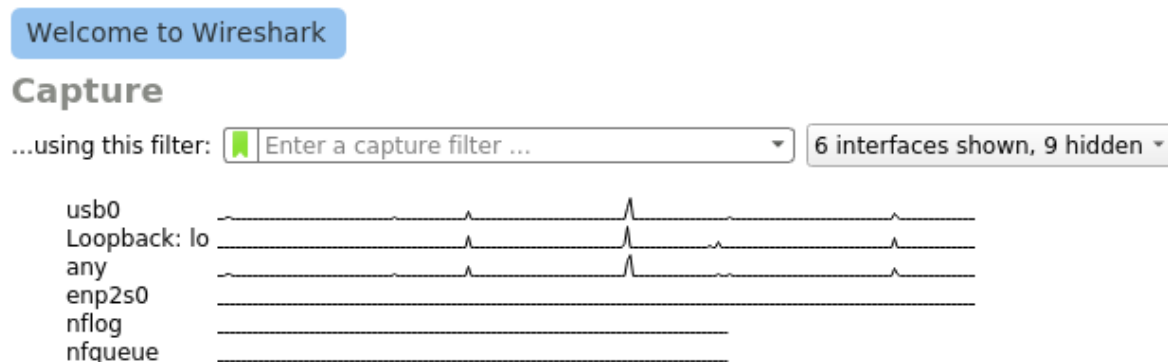


Now we will capture packages using Wireshark.

When you start Wireshark, you will see a list of interfaces that you can capture packets to and from.



There are many types of interfaces you can monitor using Wireshark, for example, **Wired**, **Wireless**, USB and many external devices. You can choose to show specific types of interfaces in the welcome screen from the marked section of the screenshot below.



Now to start capturing packets, just select the interface (in my case interface ens33) and click on the Start capturing packets icon as marked in the screenshot below.

You can also capture packets to and from multiple interfaces at the same time. Just press and hold <Ctrl> and click on the interfaces that you want to capture packets to and from and then click on the Start capturing packets icon as marked in the screenshot below.

I pinged google.com from the terminal and many packets were captured.

**Capturing from usb0**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.42.162	69.171.250.15	TLSv1.2	275	Application Data
2	0.101091597	69.171.250.15	192.168.42.162	TCP	66	443 → 49158 [ACK] Seq=
3	0.265242321	69.171.250.15	192.168.42.162	TLSv1.2	96	Application Data
4	0.308505253	192.168.42.162	69.171.250.15	TCP	66	49158 → 443 [ACK] Seq=
5	4.013797476	192.168.42.162	69.171.250.15	TLSv1.2	98	Application Data
6	4.098051222	69.171.250.15	192.168.42.162	TCP	66	443 → 49158 [ACK] Seq=
7	4.253193410	69.171.250.15	192.168.42.162	TLSv1.2	94	Application Data
8	4.253256681	192.168.42.162	69.171.250.15	TCP	66	49158 → 443 [ACK] Seq=

Frame 1: 275 bytes on wire (2200 bits), 275 bytes captured (2200 bits) on interface usb0, id 0  
 Ethernet II, Src: 8a:40:e3:e5:f9:14 (8a:40:e3:e5:f9:14), Dst: ea:d1:f7:4e:11:1d (ea:d1:f7:4e:11:1d)  
 Internet Protocol Version 4, Src: 192.168.42.162, Dst: 69.171.250.15  
 Transmission Control Protocol, Src Port: 49158, Dst Port: 443, Seq: 1, Ack: 1, Len: 209  
 Transport Layer Security

```

0000  ea d1 f7 4e 11 1d 8a 40 e3 e5 f9 14 08 00 45 00  ...N...@.....E.
0010  01 05 c6 e5 40 00 40 06 48 08 c0 a8 2a a2 45 ab  ...@...H...*.E.
0020  fa 0f c0 06 01 bb 41 18 a9 37 c1 3d da 0b 80 18  ....A...7...=...
0030  0b 44 3d b7 00 00 01 01 08 0a 82 df 46 7a 65 b3  .D=.....Fze...
0040  7c 16 17 03 03 00 cc 6f 9f 16 68 32 15 98 7a a8  |.....o...h2...z.
0050  ed f3 c4 62 a7 a5 69 fe 89 1f 9a d2 de e5 e5 34  ...b...i...4...
0060  75 2a fe 97 ce bb ea 25 27 7f 8c 4c d0 97 64 c9  u*....%'.L..d.
0070  6e f1 73 eb d7 e6 88 d1 c4 14 6e 35 c6 9e 29 12  n.s....n5...).
0080  dc 73 5f 07 8f fd 74 0f 52 69 ef 56 cf 82 64 b7  .s...t.Ri.V..d.
0090  77 69 8f c3 1b 91 06 e1 cf 16 b8 89 47 d4 6f 8c  wi.....G..o.
  
```

usb0: <live capture in progress>      Packets: 426 · Displayed: 426 (100.0%)      Profile: Default

Now you can click on a packet to select it. Selecting a packet would show many information about that packet. As you can see, information about different layers of TCP/IP Protocol is listed.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns.a == 192.168.42.8

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.42.8	69.171.250.15	TLSv1.2	98	Application Data
2	0.244018322	69.171.250.15	192.168.42.8	TLSv1.2	94	Application Data
3	0.244087699	192.168.42.8	69.171.250.15	TCP	66	57458 → 443 [ACK] Seq=33 Ac
4	0.753736331	0a:5a:49:08:3f:ca	2e:1a:92:df:2c:ff	ARP	42	Who has 192.168.42.8? Tell
5	0.753773641	2e:1a:92:df:2c:ff	0a:5a:49:08:3f:ca	ARP	42	192.168.42.8 is at 2e:1a:92
6	9.795847534	192.168.42.8	172.217.24.162	TLSv1.2	105	Application Data
7	9.796240021	192.168.42.8	172.217.24.162	TLSv1.2	90	Application Data
8	9.796291048	192.168.42.8	172.217.24.162	TCP	66	37748 → 443 [ACK] Seq=

Frame 2: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface usb0, id 0  
 Ethernet II, Src: 0a:5a:49:08:3f:ca (0a:5a:49:08:3f:ca), Dst: 2e:1a:92:df:2c:ff (2e:1a:92:df:2c:ff)  
 Internet Protocol Version 4, Src: 69.171.250.15, Dst: 192.168.42.8  
 Transmission Control Protocol, Src Port: 443, Dst Port: 57458, Seq: 1, Ack: 33, Len: 28  
 Transport Layer Security

You can also see the RAW data of that particular packet.

```
0000 aa 18 c8 07 4c 67 0a 5a 49 08 3f ca 08 00 45 00  ....Lg.Z I.?...E.
0010 05 a0 42 df 00 00 78 06 36 31 ca 86 0e 4c c0 a8  ..B...x. 61...L..
0020 2a cd 01 bb 97 24 ad 05 2a 4d f3 53 7f 9a 80 10  *....$.  *M.S....
0030 01 49 99 0a 00 00 01 01 08 0a a3 98 63 f0 83 de  .I.....c...
0040 dd f6 c9 1f 36 c8 bb ef 79 7c 18 26 89 05 9d 5b  ...6...y|.&...[
0050 4f 3e a4 63 cb aa c1 80 de bc ab a5 87 b8 60 19  0>.c.....
```

You can also click on the arrows to expand packet data for a particular TCP/IP Protocol Layer.

The screenshot shows the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for packet capture and analysis. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The packet list pane displays a table of captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
10	0.000149744	192.168.42.205	202.134.14.76	TCP	66	38692 → 443 [ACK] Seq
11	0.000188113	202.134.14.76	192.168.42.205	TCP	1454	[TCP Previous segment
12	0.000199165	192.168.42.205	202.134.14.76	TCP	78	[TCP Window Update] 3
13	0.003661509	202.134.14.76	192.168.42.205	TCP	1454	[TCP Retransmission]
14	0.003694399	192.168.42.205	202.134.14.76	TCP	66	38692 → 443 [ACK] Seq
15	0.003661894	202.134.14.76	192.168.42.205	TCP	1454	[TCP segment of a rea
16	0.003715937	192.168.42.205	202.134.14.76	TCP	66	38692 → 443 [ACK] Seq
17	0.003718670	202.134.14.76	192.168.42.205	TCP	1454	[TCP segment of a rea

The packet details pane for packet 11 shows the following structure:

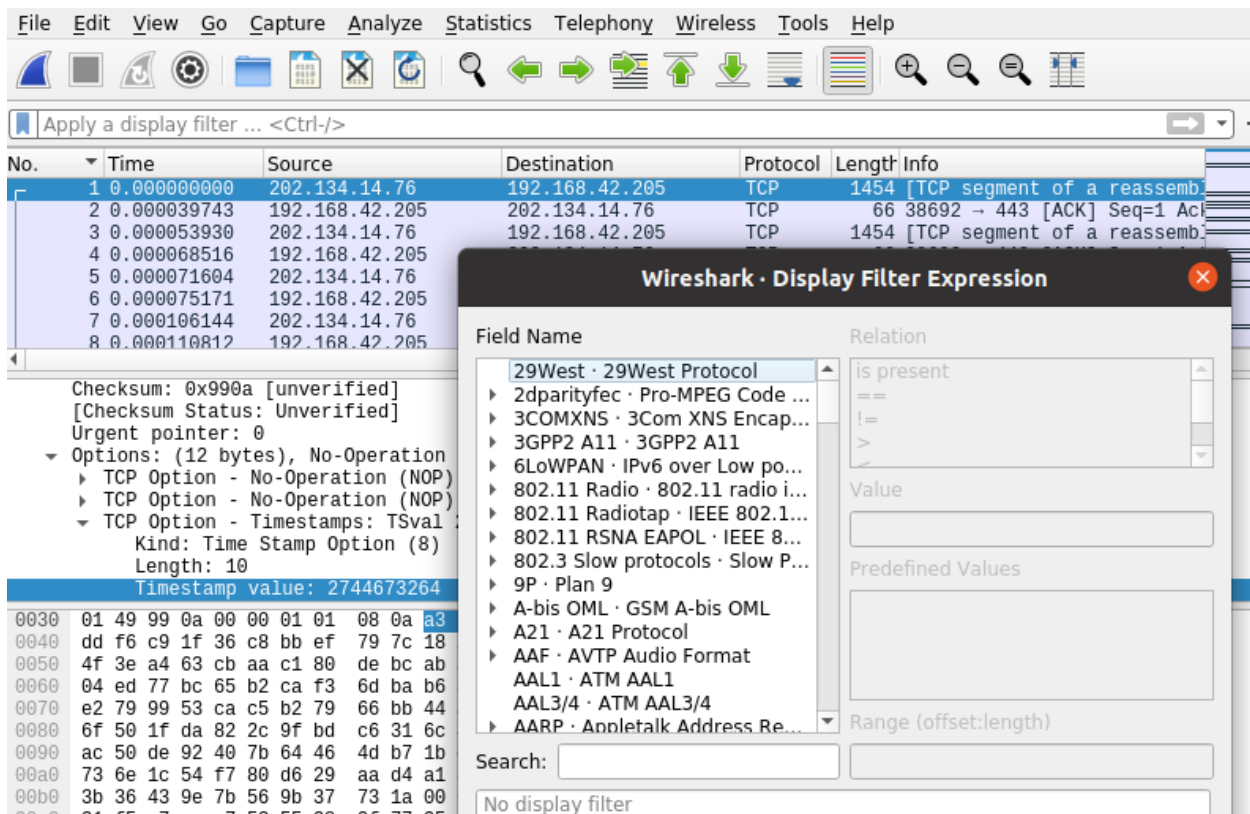
- Frame 1: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits) on interface usb0, id
- Ethernet II, Src: 0a:5a:49:08:3f:ca (0a:5a:49:08:3f:ca), Dst: aa:18:c8:07:4c:67 (aa:18:c8:07:4c:67)
- Internet Protocol Version 4, Src: 202.134.14.76, Dst: 192.168.42.205
- Transmission Control Protocol, Src Port: 443, Dst Port: 38692, Seq: 1, Ack: 1, Len: 1388
- Transport Layer Security

The packet bytes pane shows the raw data of the selected packet (packet 11) in hexadecimal and ASCII format. The ASCII column shows the text '0>.c.....'.

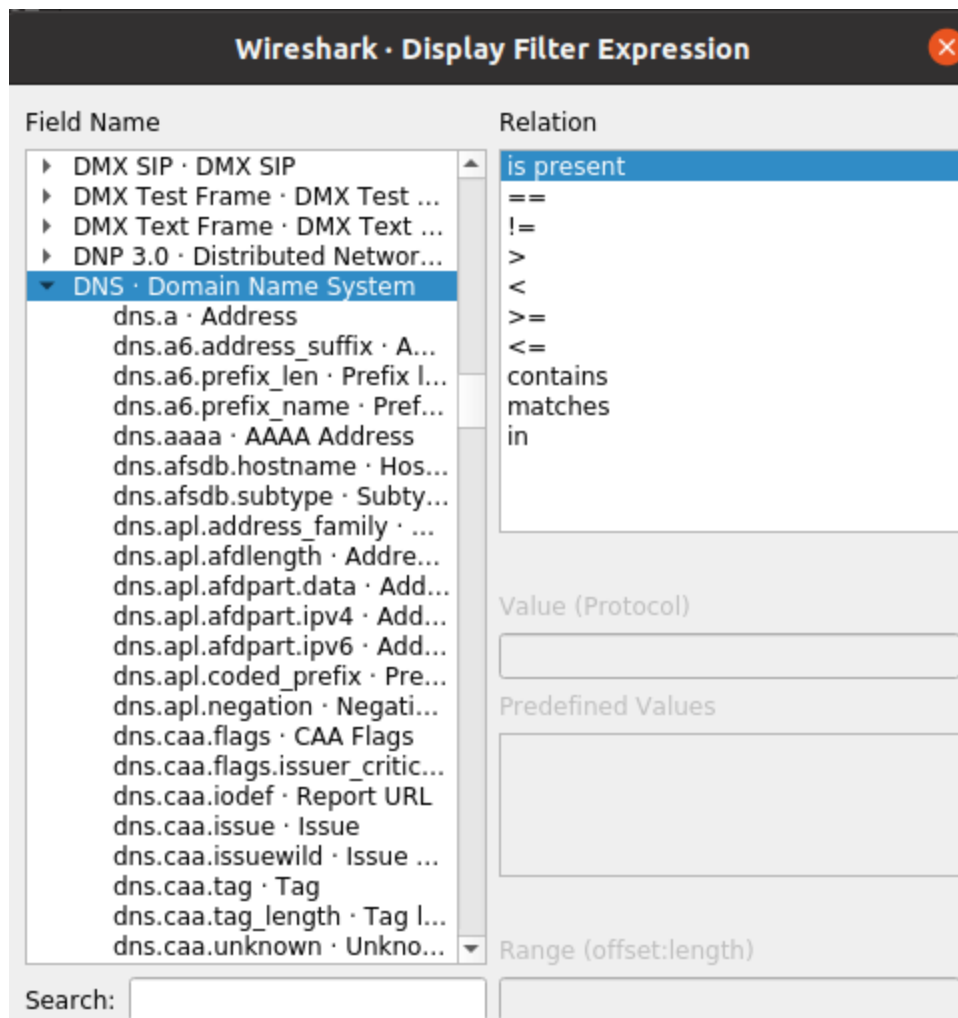
To filter packets, you can directly type in the filter expression in the textbox as marked in the screenshot below.

A new window should open as shown in the screenshot below. From here you can create filter expression to search packets very specifically.

In the **Field Name** section almost all the networking protocols are listed. The list is huge. You can type in what protocol you're looking for in the **Search** textbox and the **Field Name** section would show the ones that matched.



I am going to filter out all the DNS packets. So I selected **DNS Domain Name System** from the **Field Name** list. You can also click on the arrow on any protocol.



You can also use relational operators to test whether some field is equal to, not equal to, great than or less than some value. I searched for all the **DNS IPv4** address which is equal to **192.168.42.8** as you can see in the screenshot below.



Wireshark · Display Filter Expression

Field Name

DMX SIP · DMX SIP
DMX Test Frame · DMX Test ...
DMX Text Frame · DMX Text ...
DNP 3.0 · Distributed Networ...
DNS · Domain Name System

dns.a · Address

dns.a6.address\_suffix · A...

dns.a6.prefix\_len · Prefix l...

dns.a6.prefix\_name · Pref...

dns.aaaa · AAAA Address

dns.afsdb.hostname · Hos...

dns.afsdb.subtype · Subty...

dns.apl.address\_family · ...

dns.apl.afdlength · Addre...

dns.apl.afdpart.data · Add...

dns.apl.afdpart.ipv4 · Add...

dns.apl.afdpart.ipv6 · Add...

dns.apl.coded\_prefix · Pre...

dns.apl.negation · Negati...

dns.caa.flags · CAA Flags

dns.caa.flags.issuer\_critic...

dns.caa.iodef · Report URL

dns.caa.issue · Issue

dns.caa.issuewild · Issue ...

dns.caa.tag · Tag

dns.caa.tag\_length · Tag l...

dns.caa.unknown · Unkno...

Relation

is present

==

!=

>

<

>=

<=

contains

matches

in

Value (IPv4 address)

192.168.42.8

Predefined Values

Range (offset:length)

Search:

dns.a == 192.168.42.8

Click OK to insert this filter

OK

Cancel

Help

As you can see, only the DNS protocol packets are shown.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns.a == 192.168.42.8

No.	Time	Source	Destination	Protocol	Length	Info
13	9.927903599	192.168.42.8	172.217.24.162	TCP	66	37748 → 443 [ACK] Seq=65 Ack=2
14	11.026904233	192.168.42.8	69.171.250.15	TLSv1.2	98	Application Data
15	11.287377328	69.171.250.15	192.168.42.8	TLSv1.2	94	Application Data
16	11.287446431	192.168.42.8	69.171.250.15	TCP	66	57458 → 443 [ACK] Seq=65 Ack=57
17	14.992571983	192.168.42.8	91.189.91.157	NTP	90	NTP Version 4, client
18	15.334625363	91.189.91.157	192.168.42.8	NTP	90	NTP Version 4, server
19	15.643008205	192.168.42.8	103.242.21.17	TLSv1.2	105	Application Data
20	15.706055689	103.242.21.17	192.168.42.8	TCP	66	443 → 40612 [ACK] Seq=1 Ack=40

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface usb0, id 0

- Ethernet II, Src: 2e:1a:92:df:2c:ff (2e:1a:92:df:2c:ff), Dst: 0a:5a:49:08:3f:ca (0a:5a:49:08:3f:ca)
- Internet Protocol Version 4, Src: 192.168.42.8, Dst: 69.171.250.15
- Transmission Control Protocol, Src Port: 57458, Dst Port: 443, Seq: 1, Ack: 1, Len: 32
- Transport Layer Security

```

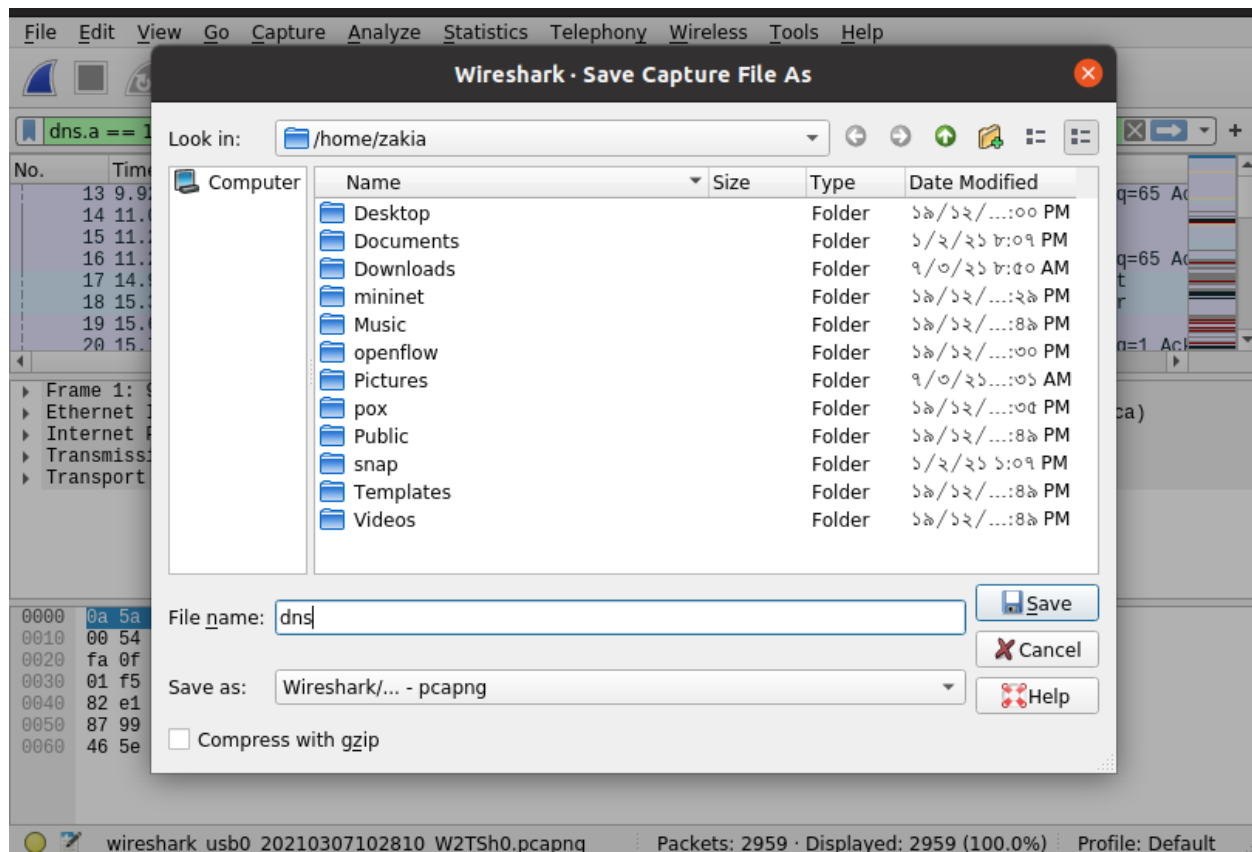
0000  0a 5a 49 08 3f ca 2e 1a 92 df 2c ff 08 00 45 00  .ZI.?.. ..E.
0010  00 54 fb 80 40 00 40 06 14 b8 c0 a8 2a 08 45 ab  .T..@.. ..E.
0020  fa 0f e0 72 01 bb 4a 4b 6a 90 e6 f4 02 dd 80 18  .r..JK j.....
0030  01 f5 ad 0e 00 00 01 01 08 0a 35 9a 52 bf cf bd  .....5.R...
0040  82 e1 17 03 03 00 1b 23 f2 24 93 51 9d f8 cd 0e  .....#$.Q....
0050  87 99 5d 22 70 e6 2d 6b 37 49 a0 f8 2b 68 4e 98  ..]"p-k 7I..+hN.
0060  46 5e                                     F^

```

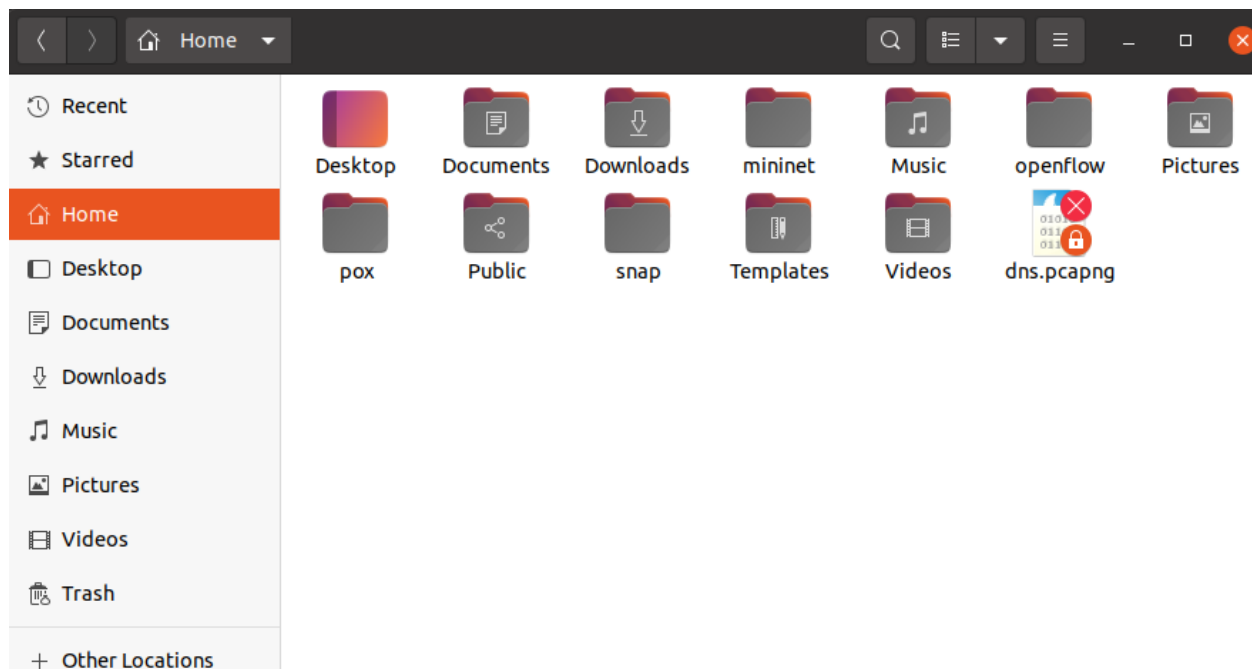
You can click on the red icon as red marked in the screenshot below to stop capturing Wireshark packets.

You can click on the saved marked icon to save captured packets to a file for future use.

Now select a destination folder, type in the file name and click on **Save**.



The file should be saved.



That's how you install and use Wireshark in Linux.

**Conclusion:** WireShark - a network protocol analyzer For Windows and \*nix systems. And although I never used it on anything but my Windows PC, it's a really interesting tool to have installed - both for developers and curious minds. So what are

some uses that might potentially benefit the people who decided to install it? Personally I have three main reasons, and i am describing those below.

### **Web debugging**

Have you ever had issues controlling the traffic flow - maybe you were calling a service but weren't sure that the requests went the right way? What about malformed HTTP requests? If any (or both) of the situations are familiar to you, then WireShark is there to help. Although it might seem that the initial returned data set is quite complicated (and trust me, there is a lot of un-needed junk captured), you can easily set specific filters to only see what you need. In many cases it reveals details I did not expect to see. Detailed data might include source port, target port, target and source IPs as well as some details about the actual physical network controller handling the processing. Packet data is also really interesting to look at, especially if it goes through SSL -WireShark has pretty decent tools to cover those details too (just look at the hex table).

### **Capture interesting stuff**

For example Windows Phone applications that come as XAP packages. Some time to set up an environment and you will be ready to intercept incoming content. Also, I found out some interesting stuff about an undocumented Zune API - also through inspecting existing transfer logs. It's really cool to see how a lot of content that is used on various web sites and application is in fact transmitted through open channels without any authentication necessary (even if that is present in the application itself). The fun fact is that you can use those channels for your own benefit (e.g. build third party clients for specific services).

### **Making sure that the right applications access the right resources**

From time to time I want to make sure that every application I use, that has access to the Internet, only accesses resources it should. WireShark pretty much

covers every transfer layer - of course, sometimes it is hard to see what data is passed between machines due to the fact that it is encrypted, but nonetheless, it is interesting to keep track at least where the HTTP traffic is targeted.

If you go through some packets and HTTP POST requests, you will be able to see what information is sent from your device to a remote server. For example, Rafael Riviera was able to track down the data transmitted from a Windows Phone 7 device to the Software Quality Management server in a similar manner.

WireShark is not that big and doesn't consume enormous quantities of resources, so it runs pretty well in the background while other processes are running. I would definitely recommend to try it out, even just for fun, to see what you can get net-wise out of it.