

**MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY**  
**Santosh, Tangail – 1902**



**Course Title: Computer Networks Lab**

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

**Lab Report No: 08**

**Submitted by,**

**Name: Md Ahadul Haque &  
Ali Ashadullah Arif**

**Submitted to,**

**NAZRUL ISLAM  
Assistant Professor**

**ID: IT-18045 & 18031**

**Session: 2017-18**

**Dept. of ICT, MBSTU**

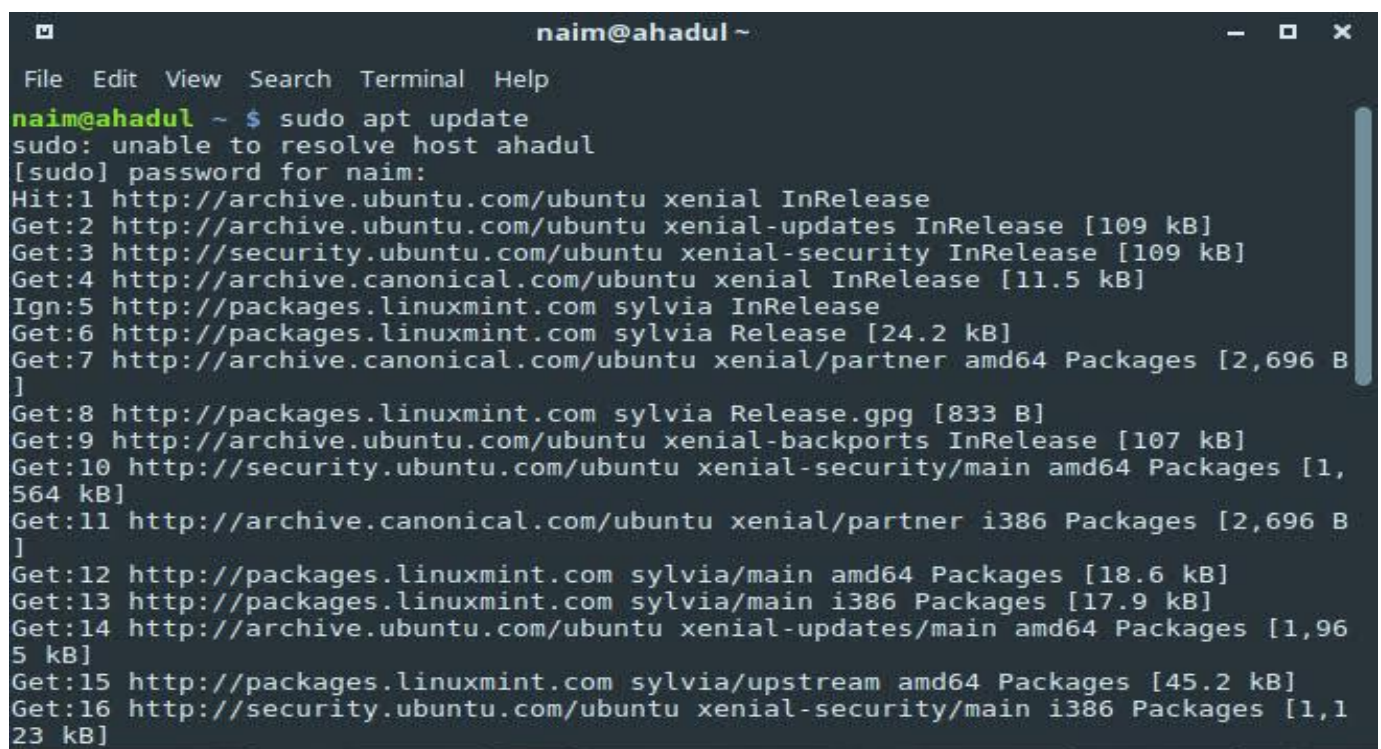
**Dept. of ICT, MBSTU**

## 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 'naim@ahadul ~' showing the output of the 'sudo apt update' command. The output lists various package repositories and their update status, including InRelease and Release files from archive.ubuntu.com, security.ubuntu.com, archive.canonical.com, and packages.linuxmint.com. The terminal shows the progress of updating the package cache for different architectures (amd64, i386) and package sets (main, universe, restricted, multiverse).

```
naim@ahadul ~ $ sudo apt update
sudo: unable to resolve host ahadul
[sudo] password for naim:
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:3 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Get:4 http://archive.canonical.com/ubuntu xenial InRelease [11.5 kB]
Ign:5 http://packages.linuxmint.com sylvia InRelease
Get:6 http://packages.linuxmint.com sylvia Release [24.2 kB]
Get:7 http://archive.canonical.com/ubuntu xenial/partner amd64 Packages [2,696 B]
]
Get:8 http://packages.linuxmint.com sylvia Release.gpg [833 B]
Get:9 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:10 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [1,564 kB]
Get:11 http://archive.canonical.com/ubuntu xenial/partner i386 Packages [2,696 B]
]
Get:12 http://packages.linuxmint.com sylvia/main amd64 Packages [18.6 kB]
Get:13 http://packages.linuxmint.com sylvia/main i386 Packages [17.9 kB]
Get:14 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [1,965 kB]
Get:15 http://packages.linuxmint.com sylvia/upstream amd64 Packages [45.2 kB]
Get:16 http://security.ubuntu.com/ubuntu xenial-security/main i386 Packages [1,123 kB]
```

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)
```

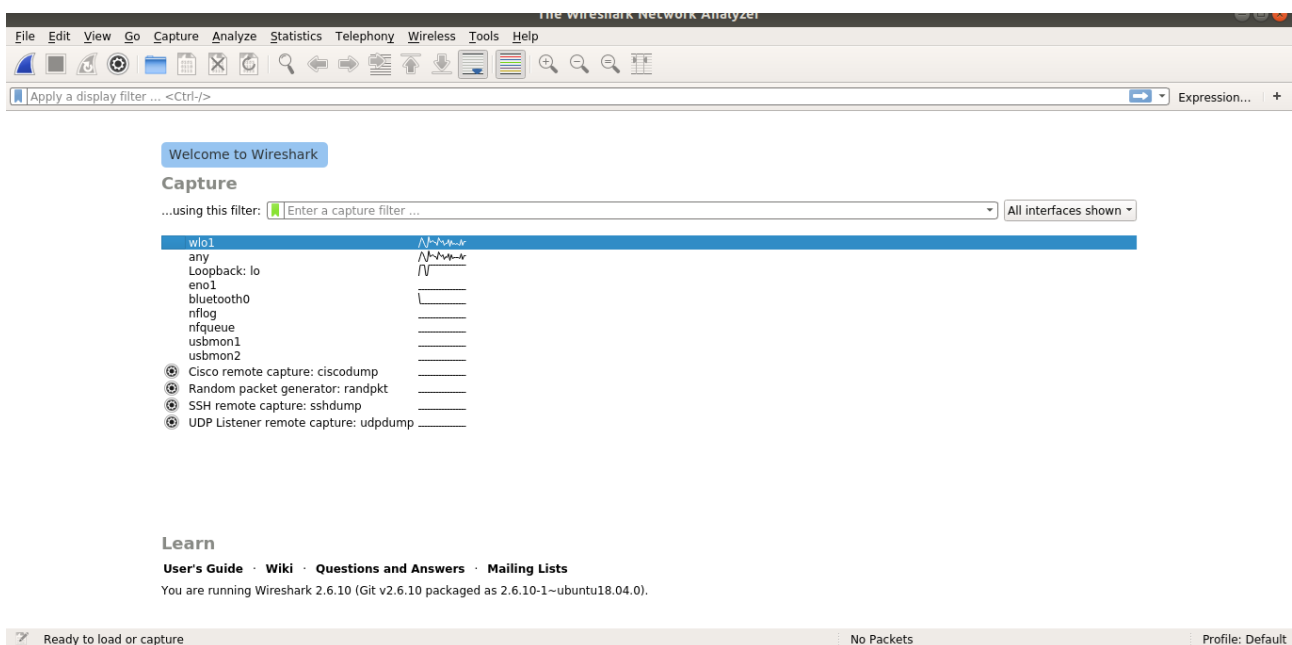
Now reboot your computer with the following command:

```
$ sudo reboot
```

Now run Wireshark using the following command:

```
$ sudo wireshark
```

```
naim@ahadul ~  
File Edit View Search Terminal Help  
naim@ahadul ~ $ sudo wireshark  
sudo: unable to resolve host ahadul  
[sudo] password for naim:  
sudo: wireshark: command not found  
naim@ahadul ~ $
```

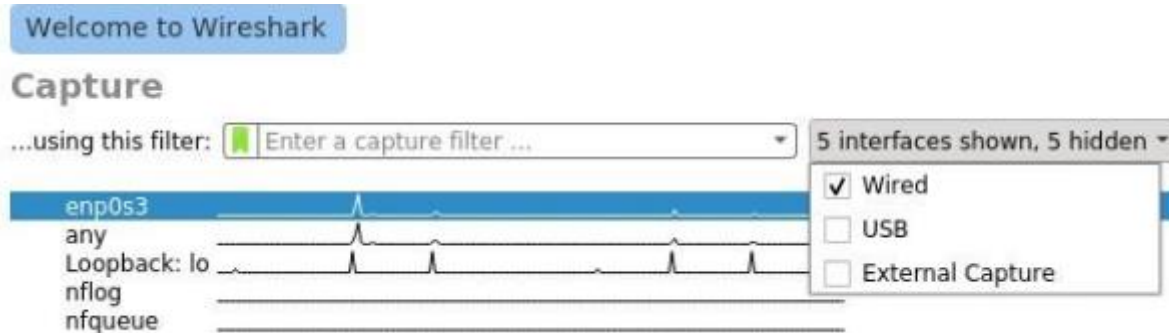


Now we will capture packages using Wireshark.

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

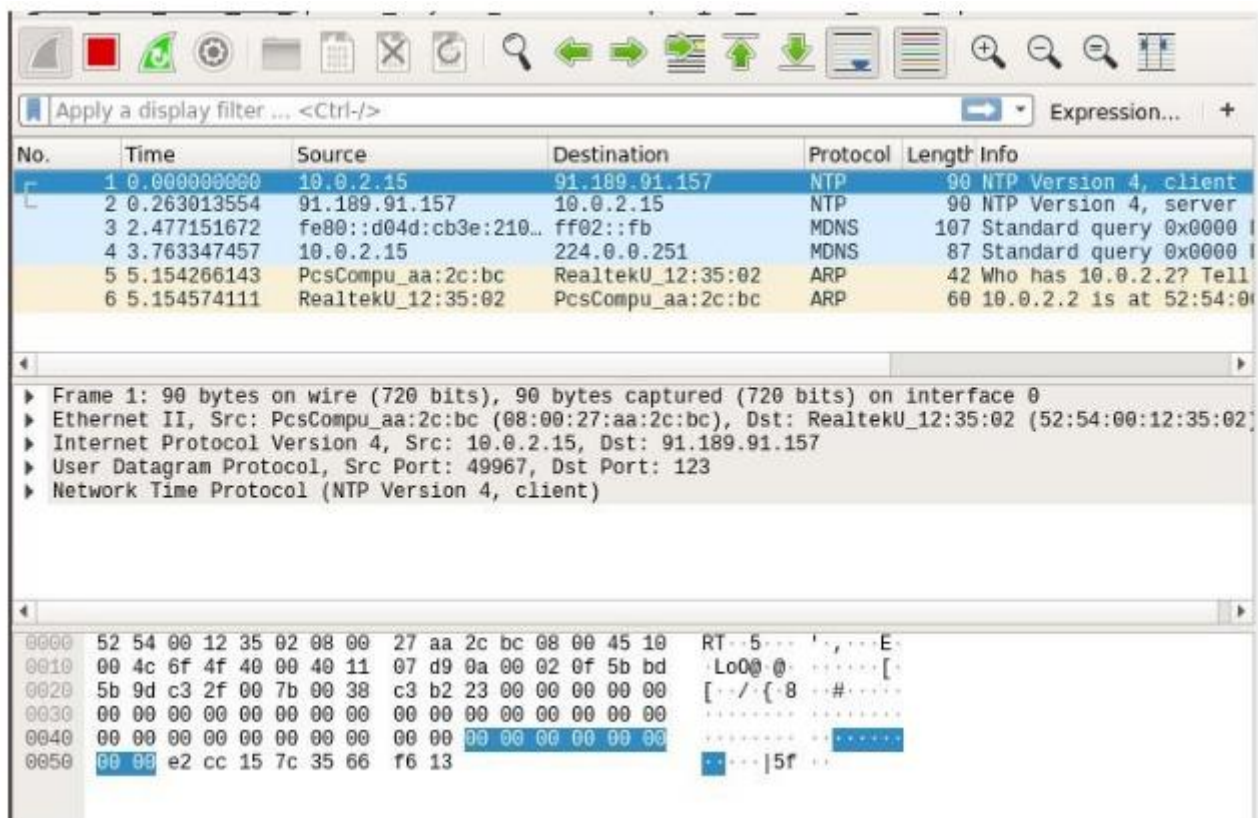


There are many types of interfaces you can monitor using Wireshark, for example, **Wired**, **Wireless**, USB and many external devices. You can choose to show specific types of interfaces in the welcome screen from the marked section of the screenshot below.

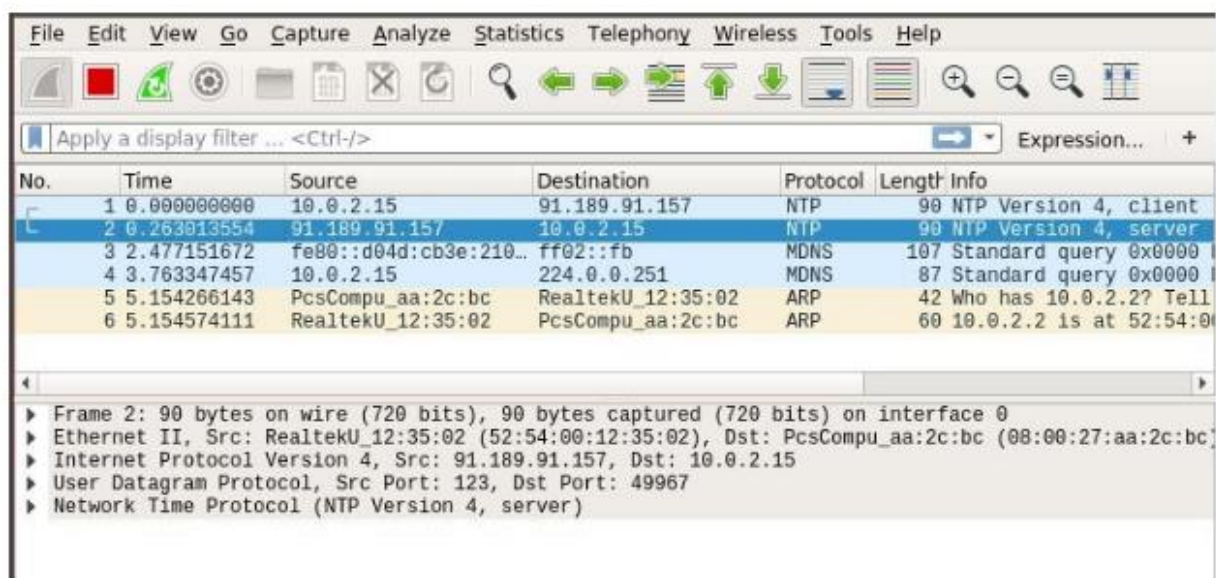


Now to start capturing packets, just select the interface (in my case interface ens33) and click on the **Start capturing packets** icon as marked in the screenshot below. You can also capture packets to and from multiple interfaces at the same time. Just press and hold **<Ctrl>** and click on the interfaces 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.



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.



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



0000	08 00 27 aa 2c bc 52 54 00 12 35 02 08 00 45 00	..',.RT..5...E
0010	00 4c 01 06 00 00 40 11 b6 32 5b bd 5b 9d 0a 00	.L...@..2[...]
0020	02 0f 00 7b c3 2f 00 38 8c e4 24 02 03 e8 00 00	...{/8...\$...
0030	0c 43 00 00 09 9d 84 a3 61 01 e2 cc 11 f7 45 fa	.C...a...E
0040	1a d4 e2 cc 15 7c 35 66 f6 13 e2 cc 15 7d 08 ff	... 5f...}
0050	0a 5d e2 cc 15 7d 09 00 41 d1	.]...}...A

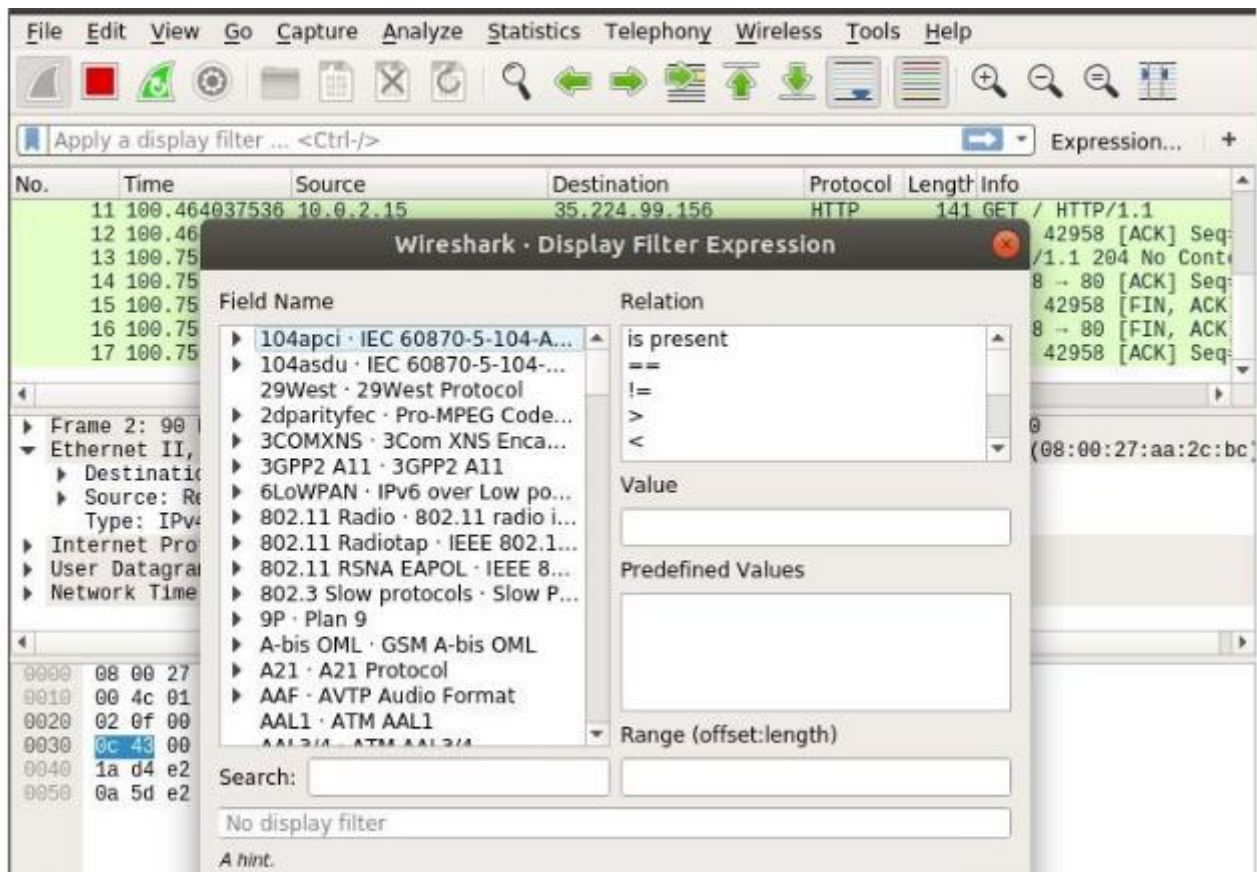
You can also click on the arrows to expand packet data for a particular TCP/IP Protocol Layer.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help								
<div>Apply a display filter ... &lt;Ctrl-/&gt; Expression... +</div>								
No.	Time	Source	Destination	Protocol	Length	Info		
11	100.464037536	10.0.2.15	35.224.99.156	HTTP	141	GET / HTTP/1.1		
12	100.464736669	35.224.99.156	10.0.2.15	TCP	60	80 → 42958 [ACK] Seq:		
13	100.755351084	35.224.99.156	10.0.2.15	HTTP	202	HTTP/1.1 204 No Cont		
14	100.755380438	10.0.2.15	35.224.99.156	TCP	54	42958 → 80 [ACK] Seq:		
15	100.755544083	35.224.99.156	10.0.2.15	TCP	60	80 → 42958 [FIN, ACK		
16	100.755875413	10.0.2.15	35.224.99.156	TCP	54	42958 → 80 [FIN, ACK		
17	100.756369423	35.224.99.156	10.0.2.15	TCP	60	80 → 42958 [ACK] Seq:		
▶ Frame 2: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0								
▼ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_aa:2c:bc (08:00:27:aa:2c:bc)								
▶ Destination: PcsCompu_aa:2c:bc (08:00:27:aa:2c:bc)								
▶ Source: RealtekU_12:35:02 (52:54:00:12:35:02)								
Type: IPv4 (0x0800)								
▶ Internet Protocol Version 4, Src: 91.189.91.157, Dst: 10.0.2.15								
▶ User Datagram Protocol, Src Port: 123, Dst Port: 49967								
▶ Network Time Protocol (NTP Version 4, server)								
0000	08 00 27 aa 2c bc 52 54	00 12 35 02 08 00 45 00	. . . , . RT . . 5 . . . E .					
0010	00 4c 01 06 00 00 40 11	b6 32 5b bd 5b 9d 0a 00	. L . . . @ . . 2 [ . . . ]					
0020	02 0f 00 7b c3 2f 00 38	8c e4 24 02 03 e8 00 00	. . . { . / 8 . . \$ . . .					
0030	0c 43 00 00 09 9d 84 a3	61 01 e2 cc 11 f7 45 fa	. C . . . a . . . E .					
0040	1a d4 e2 cc 15 7c 35 66	f6 13 e2 cc 15 7d 08 ff	. . . .   5f . . . . } . .					
0050	0a 5d e2 cc 15 7d 09 00	41 d1	. . . . } . . . A .					

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.

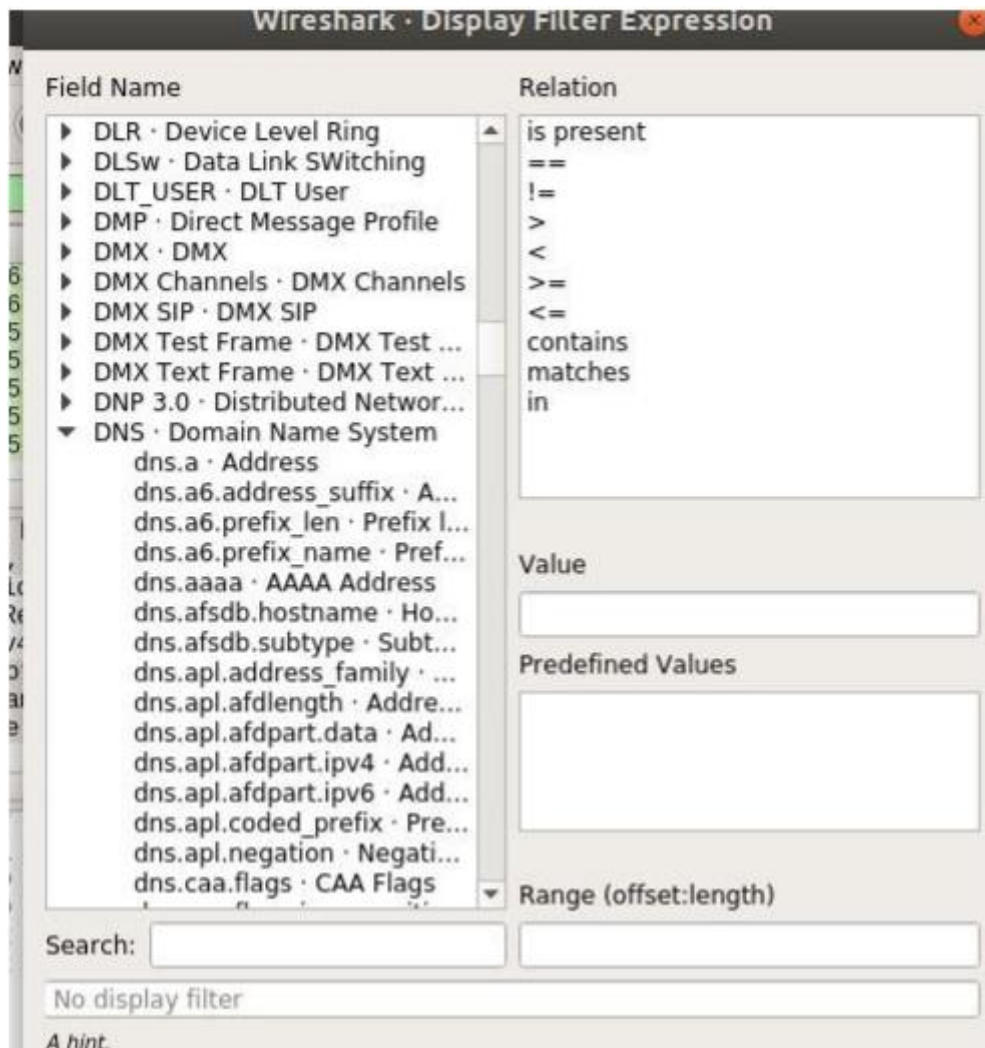
In the **Field Name** section almost all the networking protocols are listed. The list is huge. You can type in what protocol you're looking for in the **Search** textbox and the **Field Name** section would show the ones that matched.



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.

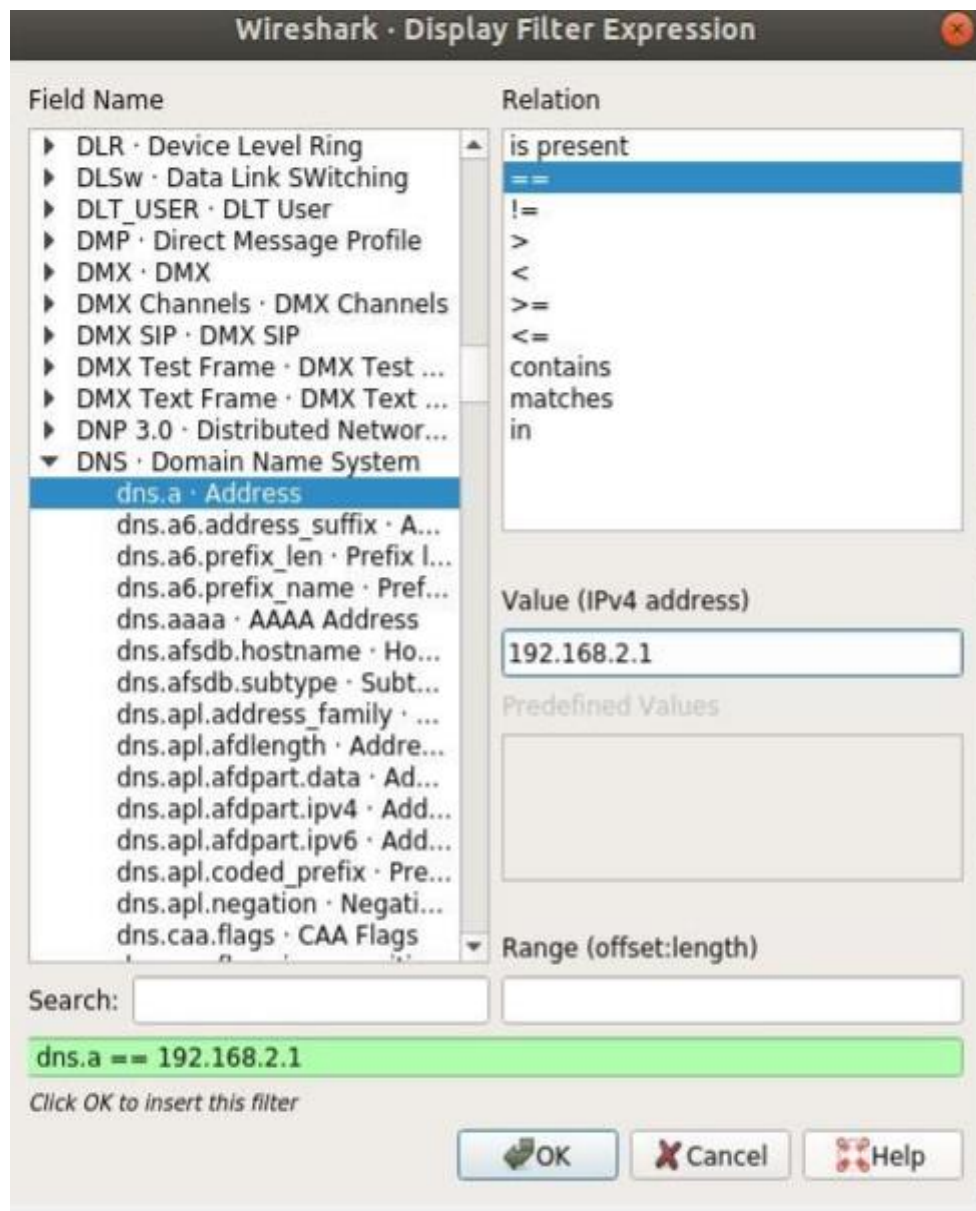
You  
also use

can

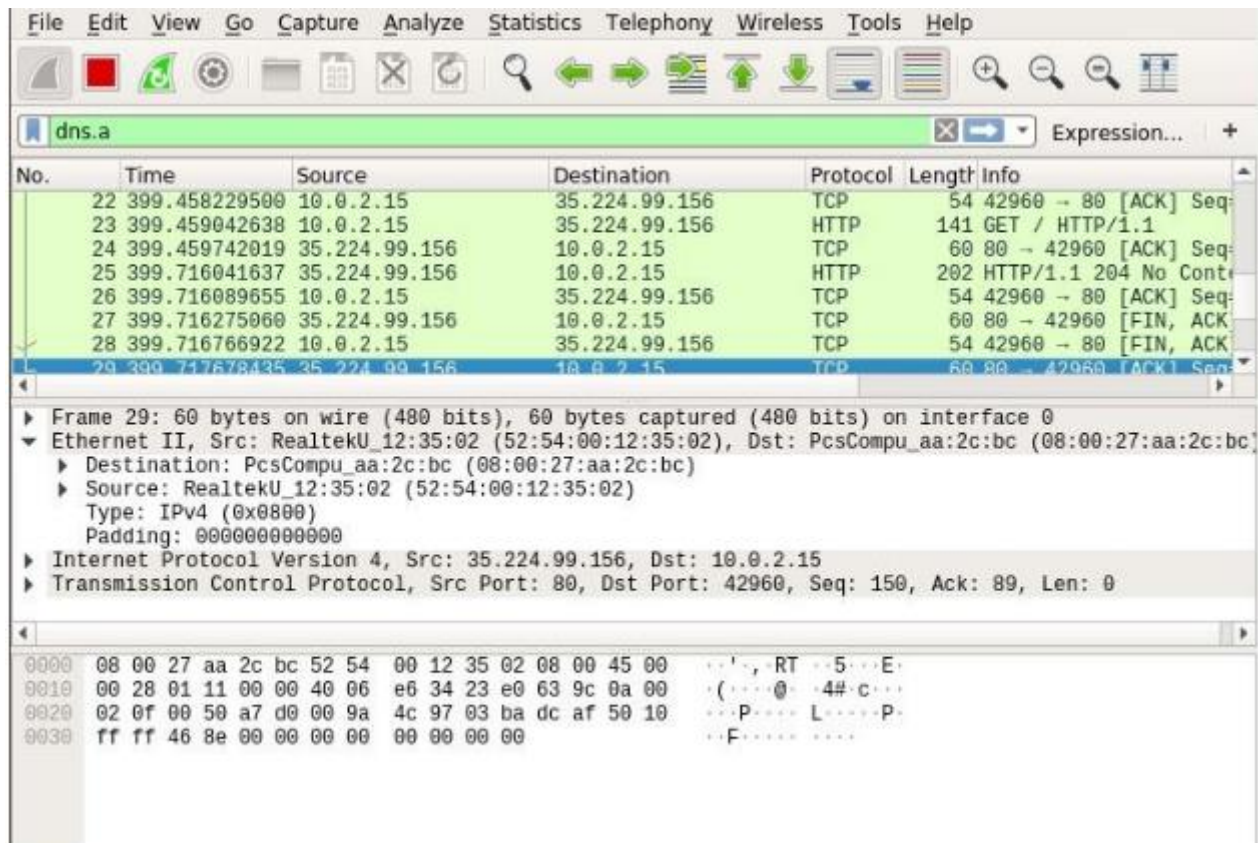


relational operators to test whether some field is equal to, not equal to, great than or less than some value. I searched for all the **DNS IPv4** address which is equal to **192.168.2.1** as you can see in the screenshot below.





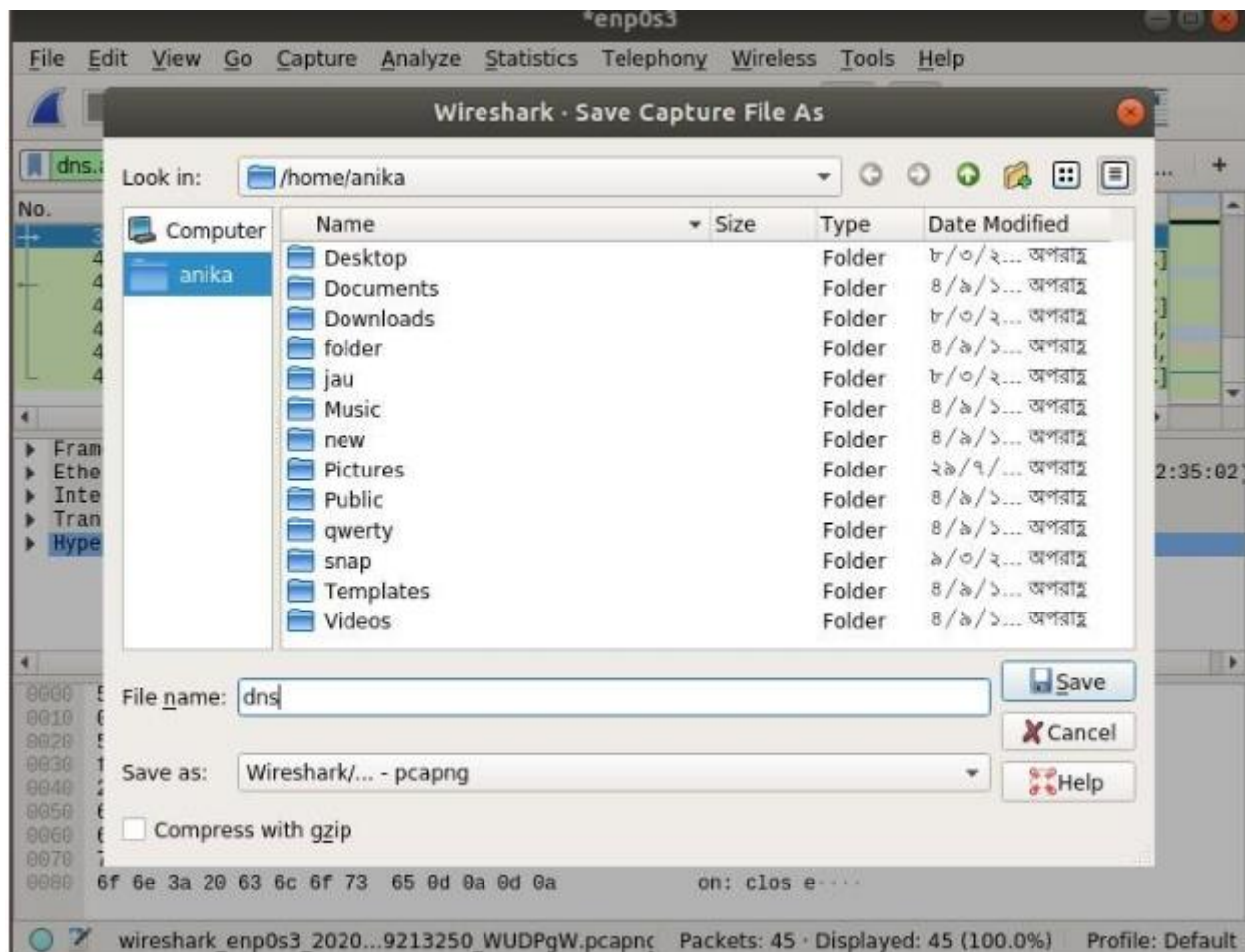
As you can see, only the DNS protocol packets are shown.



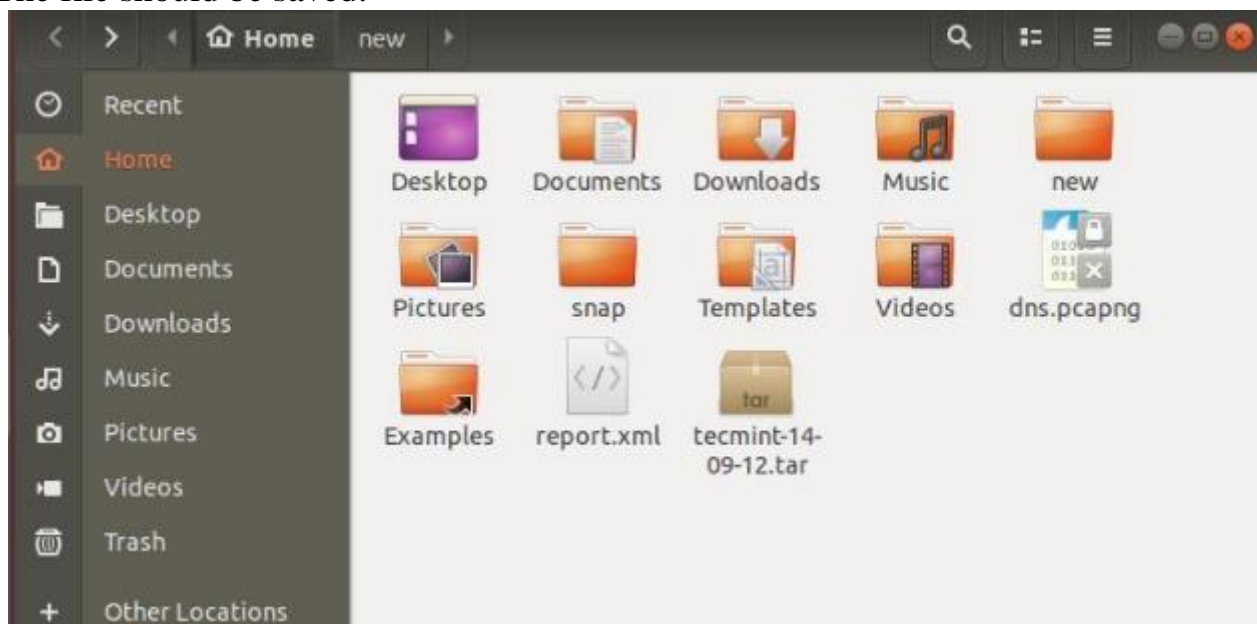
You can click on the red icon as red marked in the screenshot below to stop capturing Wireshark packets.

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.