Mawlana Bhashani Science and Technology University

Lab-Report

Report No:

Course code: ICT-3207

Course title: Computer Networks

Date of Performance:05/03/2021

Date of Submission:

Submitted by

Name: Afrin Zaman & Mahfuza Talukdar

ID:IT-18008 & IT*18009

3rd year 2nd semester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

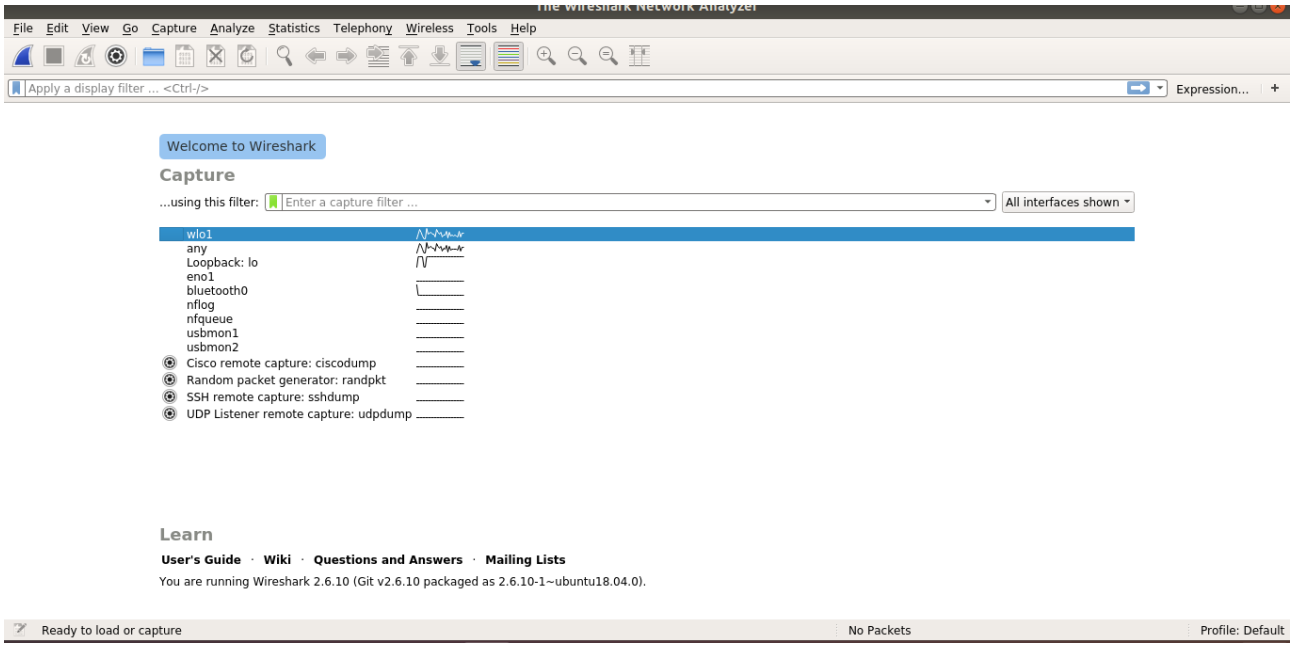
INSTALLING WIRESHARK:

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.
Sudo apt update

```
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://bd.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://bd.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://bd.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
230 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

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

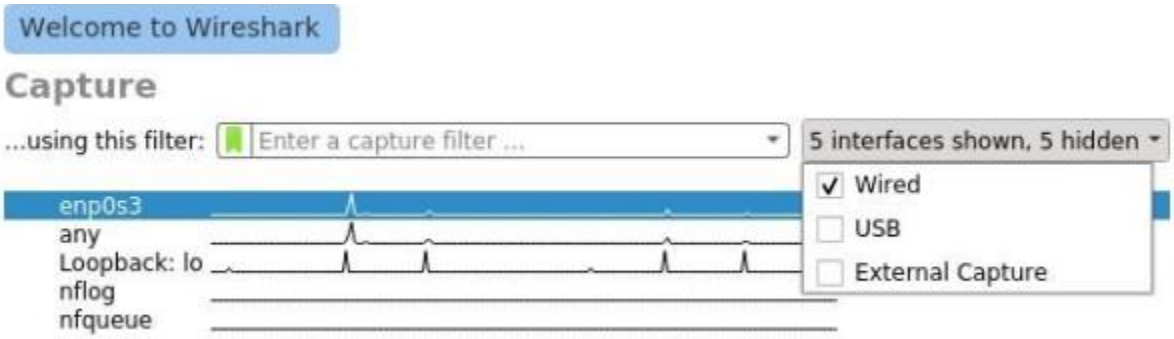
```
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
01:03:36.516         Warn Could not compile "of" in colorfilters file "/home/tazneen/.wireshark/colorfilters".
"of" is neither a field nor a protocol name.
01:03:36.516         Warn Could not compile "Checksum Errors" in colorfilters file "/home/tazneen/.wireshark/colorfilters".
Neither "cdp.checksum_bad" nor "1" are field or protocol names.
```



Now we will capture packages using Wireshark.

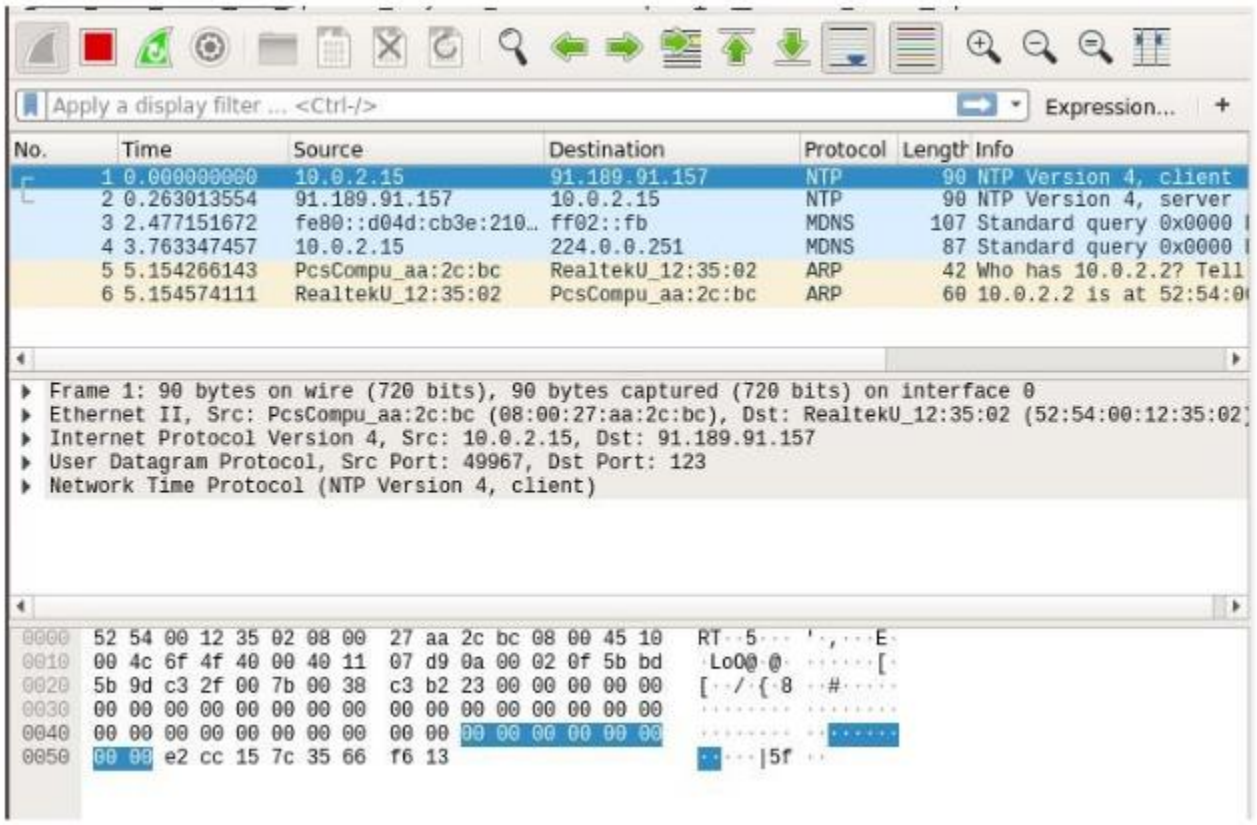
When you start Wireshark, you will see a list of interfaces that you can capture packets to and from.

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.

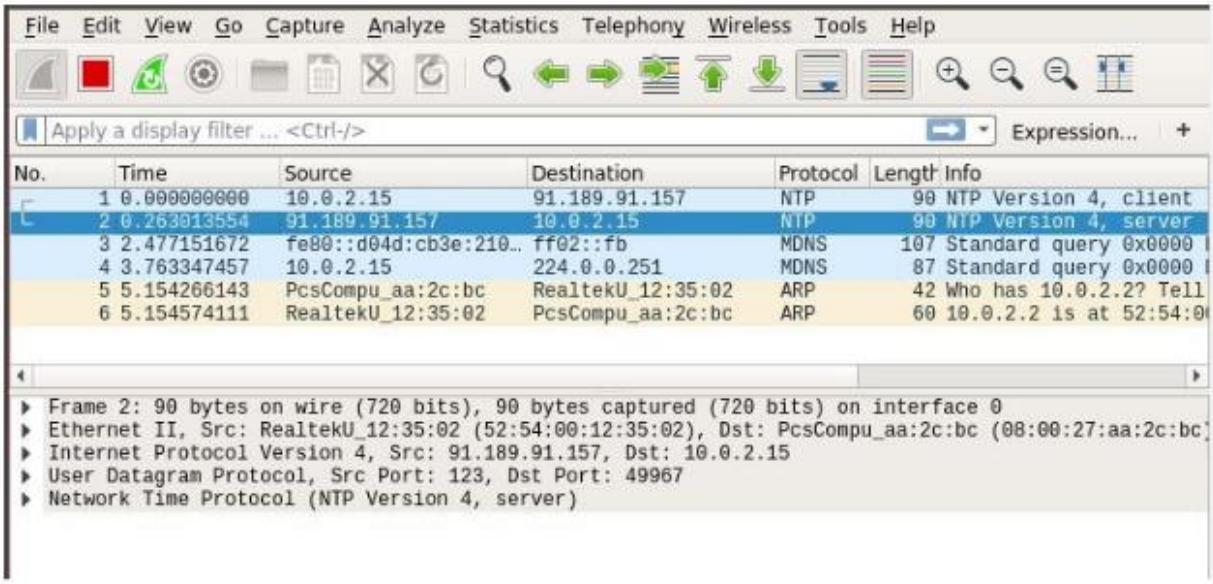


Now to start capturing packets, just select the interface (in my case interface ens33) and click on the **Start capturing packets** icon as marked in the screenshot below.
You can also capture packets to and from multiple interfaces at the same time. Just press and hold **<Ctrl>** and click on the interfaces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

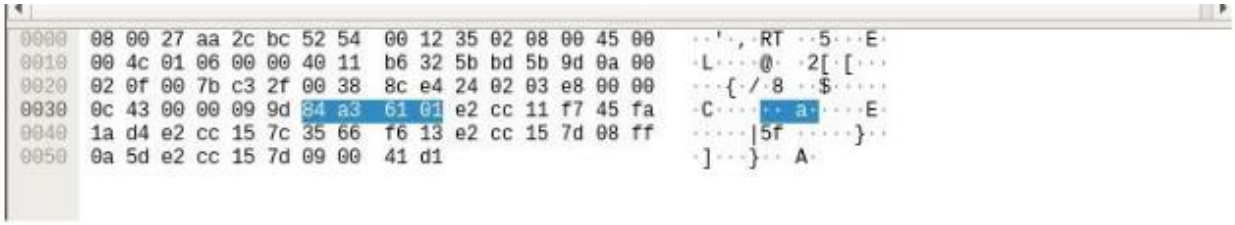
I pinged google.com from the terminal and many packets were captured.



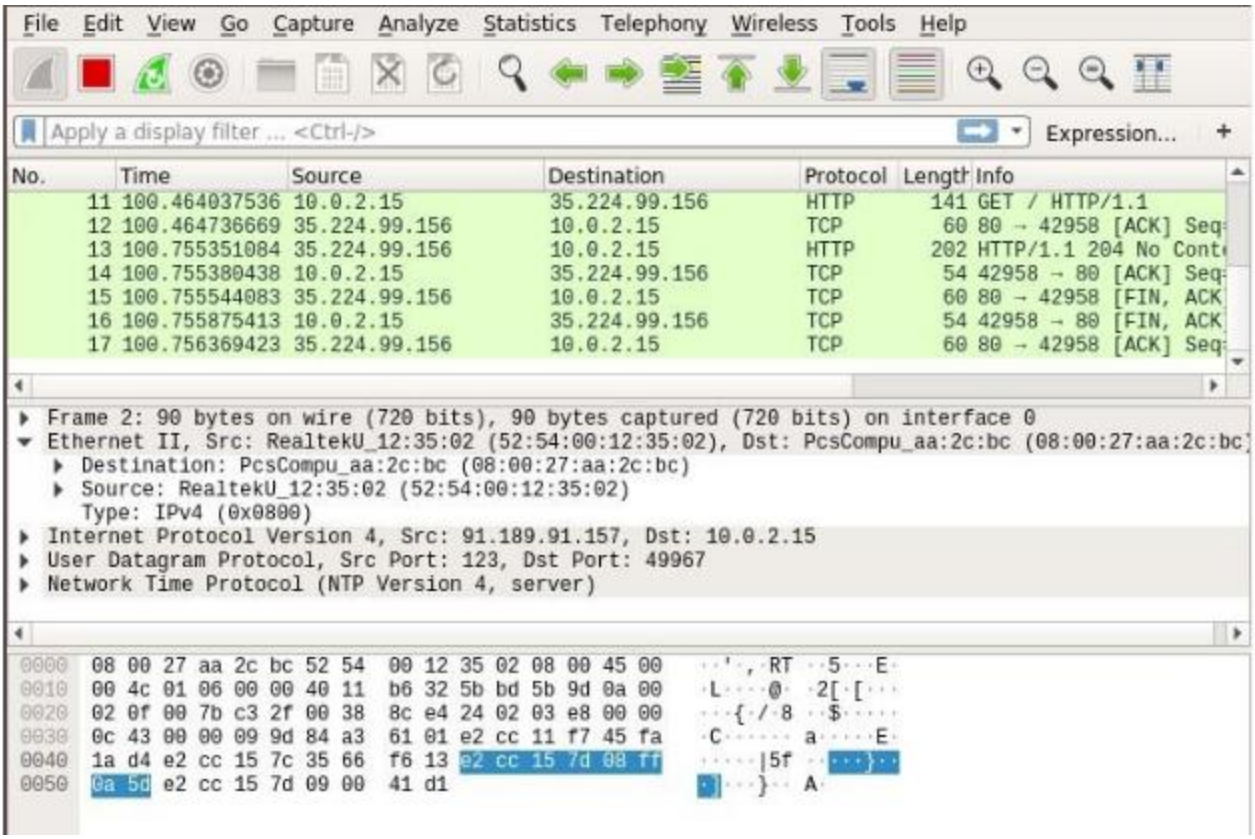
Now you can click on a packet to select it. Selecting a packet would show many information about that packet. As you can see, information about different layers of TCP/IP Protocol is listed.



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



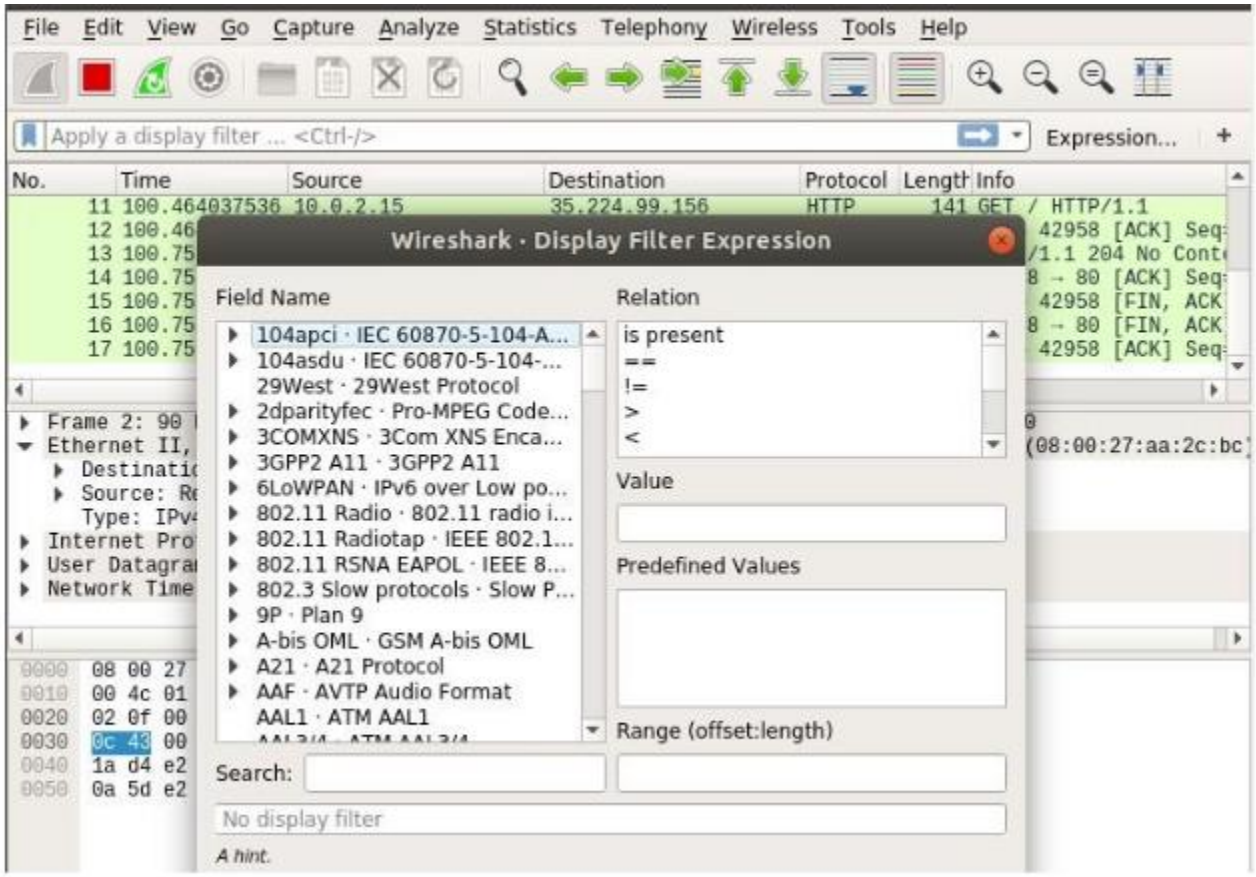
You can also click on the arrows to expand packet data for a particular TCP/IP Protocol Layer.



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.

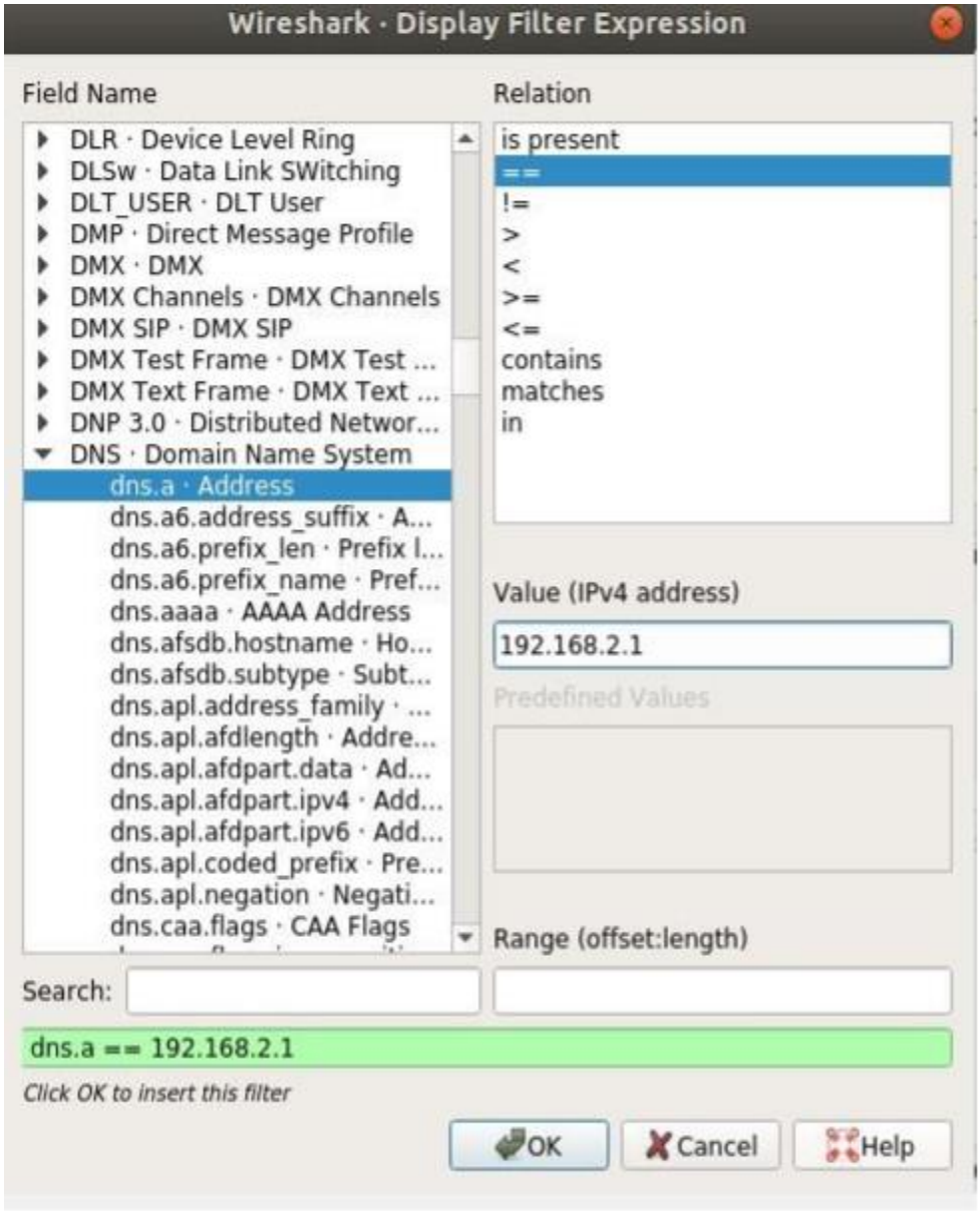
In the **Field Name** section almost all the networking protocols are listed. The list is huge. You can type in what protocol you're looking for in the **Search** textbox and the **Field Name** section would show the ones that matched.



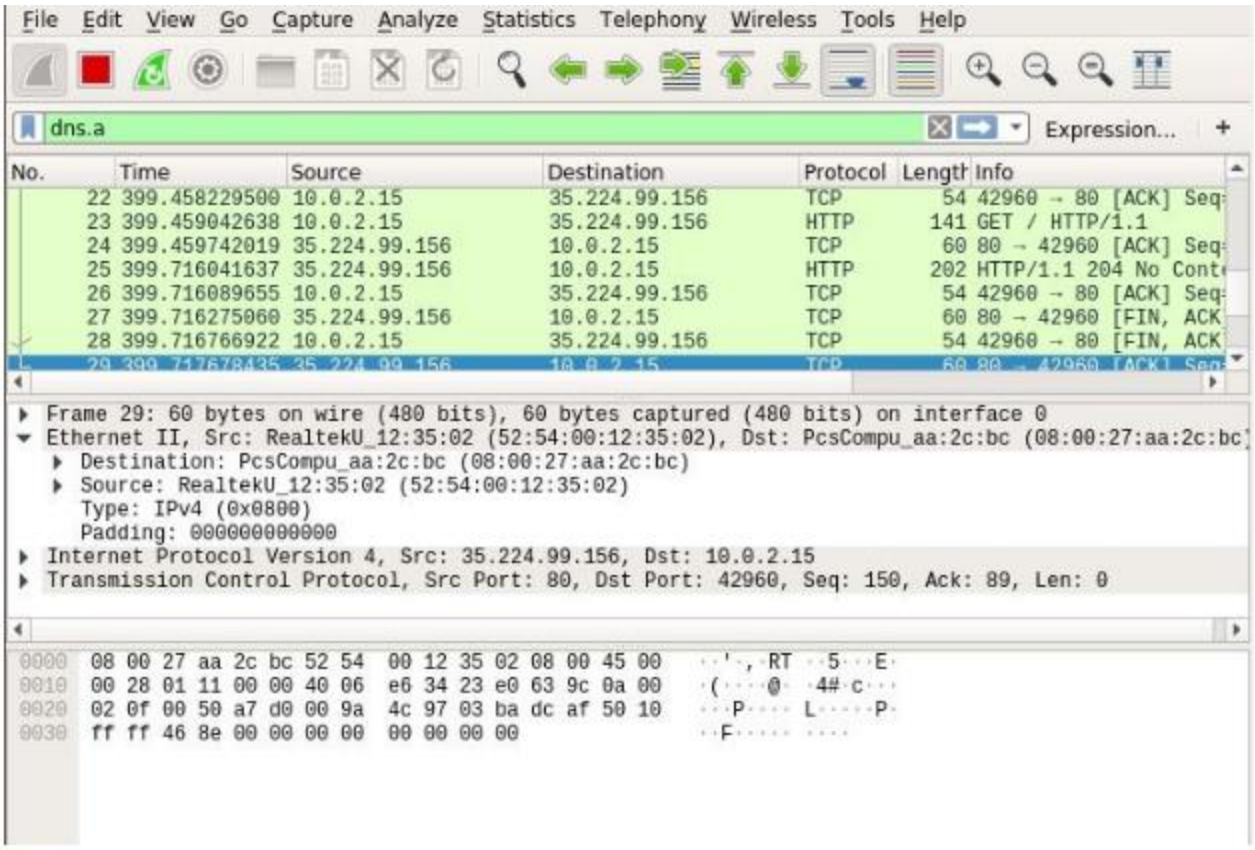
I am going to filter out all the DNS packets. So I selected **DNS Domain Name System** from the **Field Name** list. You can also click on the arrow on any protocol.

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 **192.168.2.1** as you can see in the screenshot below.



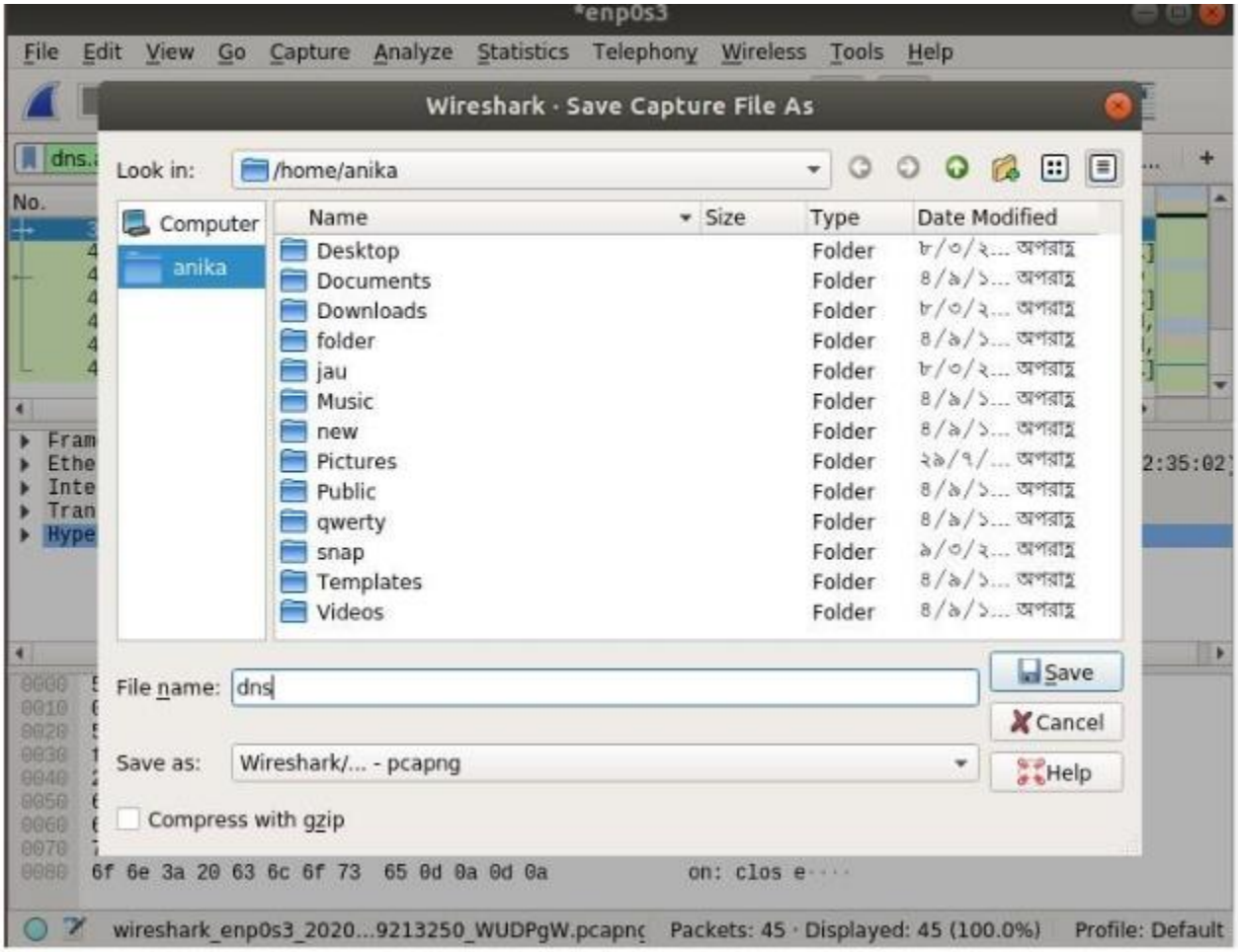
As you can see, only the DNS protocol packets are shown.



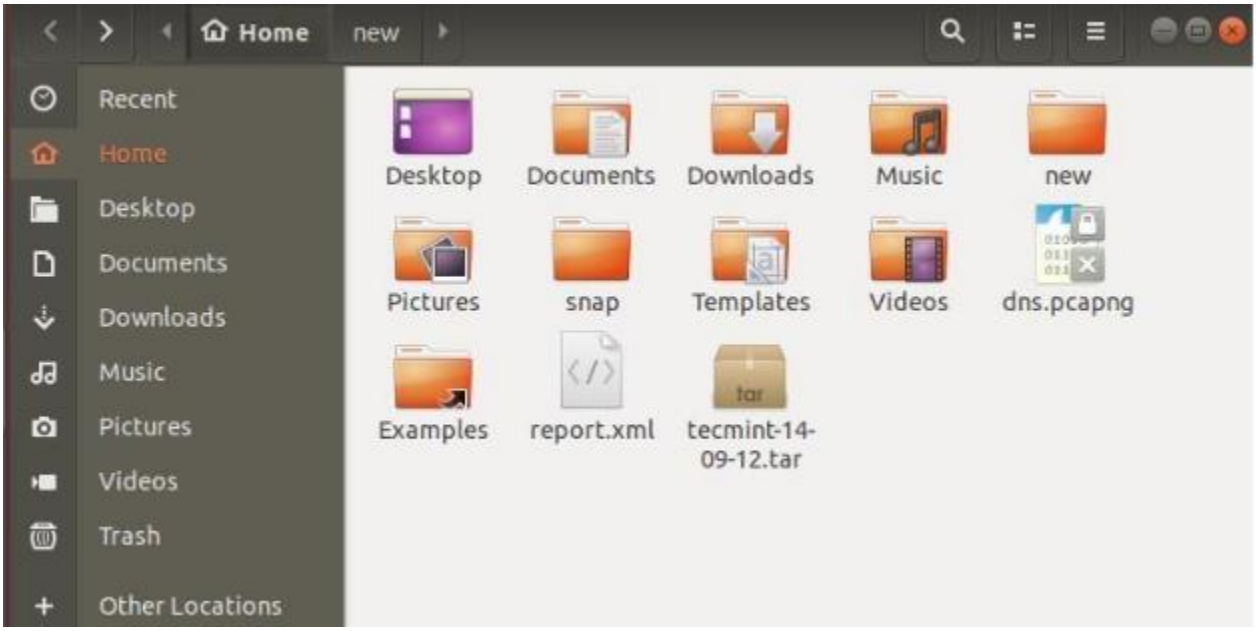
You can click on the red icon as red marked in the screenshot below to stop capturing Wireshark packets.

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.

Discussion: 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 quanitites 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.