

---

# Wireshark Activity

Irfan Kanat

06/02/2022



## Wireshark

Today's exercise is to solidify the concepts learned in the classroom. We will specifically focus on the TCP/IP 5 Layer Stack and how it looks in action.

For this purpose we will use a tool called wireshark. Wireshark is a packet sniffer. It can be used to record and inspect traffic going through your network interfaces (wireless, ethernet...).

Wireshark listens to all the packages going through the interface and records them. If you follow security news, you will see that it is often used to trace malware communication to command and control servers.

You should be careful when using this tool in organizational networks. Since there is the possibility of intercepting traffic not meant for you, you may violate the law or the organizational policy. This is why I recommend not using wireshark outside of your home network.

I include two exercises here. One simple and one a bit more advanced. If you still have time on your hands after you are done, you can just capture packages from your computer at rest and see all the background chatter originating from your machine. Who knows, may be you will catch some bad guys...

## Installation

Wireshark is an open source tool and can be freely obtained from <https://www.wireshark.org/>.

Download the appropriate version for your computer and run the downloaded file.

Installation is straightforward. Default settings are sane and you don't need to change anything. Just proceed as advised.

At the end of the installation you will need to restart your computer.

## Wireshark Analysis Simple

Click the link below to download the pcapng file. This is from a capture I made earlier. I pinged the cbs.dk server 3 times and recorded the results. It also includes the ARP, and DNS exchanges to figure out the route to cbs.dk. See [this document](#) for details on how capture was carried out.

[ping capture](#)

You can open the downloaded file in wireshark. Either double click it, or go through the File > Open dialogue.

This is what it should look like:

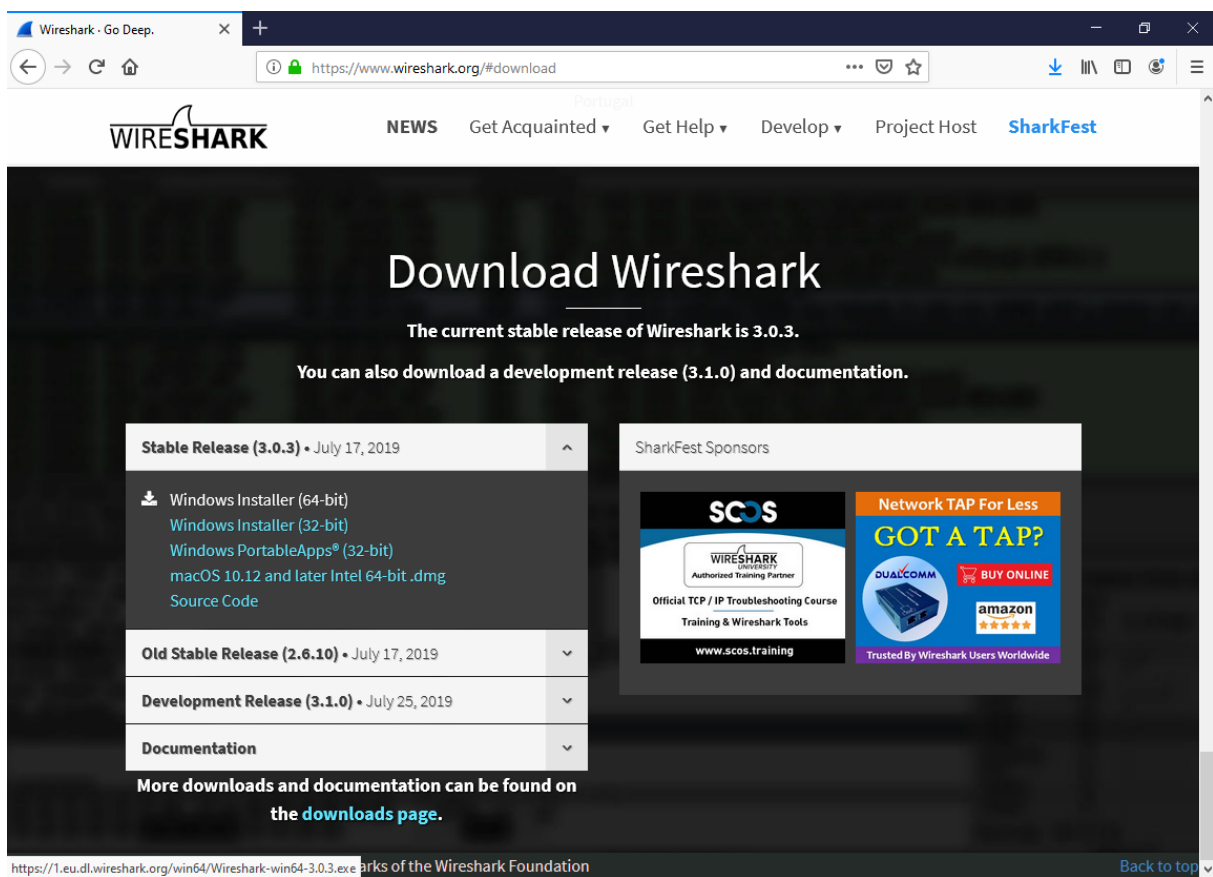
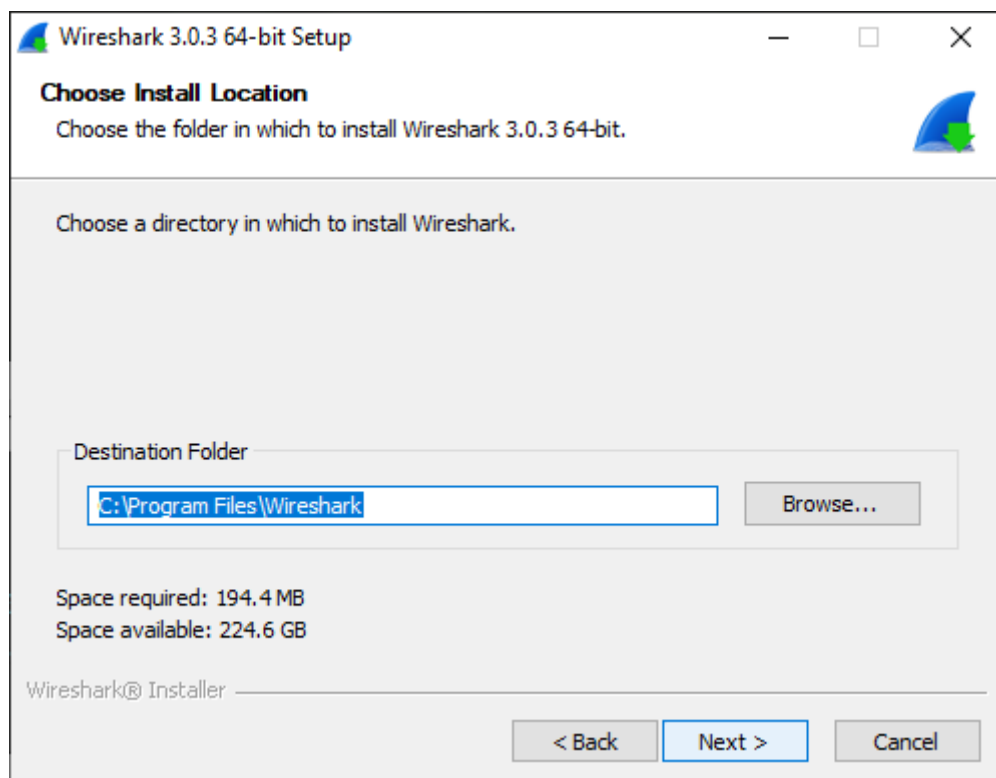


Figure 1: Wireshark.org



**Figure 2:** Installation

pingCapture.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	PcsCompu_6d:ac:0d	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
2	0.000200	RealtekU_12:35:02	PcsCompu_6d:ac:0d	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
3	0.292677	10.0.2.15	10.0.2.3	DNS	66	Standard query 0xea09 A cbs.dk
4	0.294384	10.0.2.3	10.0.2.15	DNS	82	Standard query response 0xea09 A cbs.dk A 130.226.47.28
5	0.305552	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=15/3840, ttl=128 (reply in 6)
6	0.306352	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=15/3840, ttl=127 (request in 5)
7	1.343854	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=16/4096, ttl=128 (reply in 8)
8	1.344752	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=16/4096, ttl=127 (request in 7)
9	2.359190	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=17/4352, ttl=128 (reply in 10)
10	2.360013	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=17/4352, ttl=127 (request in 9)

Epoch Time: 1564580161.349712000 seconds  
 [Time delta from previous captured frame: 0.000000000 seconds]  
 [Time delta from previous displayed frame: 0.000000000 seconds]  
 [Time since reference or first frame: 0.000000000 seconds]  
 Frame Number: 1  
 Frame Length: 42 bytes (336 bits)  
 Capture Length: 42 bytes (336 bits)  
 [Frame is marked: False]  
 [Frame is ignored: False]  
 [Protocols in frame: eth:ethertype:arp]  
 [Coloring Rule Name: ARP]  
 [Coloring Rule String: arp]

▼ Ethernet II, Src: PcsCompu\_6d:ac:0d (08:00:27:6d:ac:0d), Dst: RealtekU\_12:35:02 (52:54:00:12:35:02)  
 ▼ Destination: RealtekU\_12:35:02 (52:54:00:12:35:02)

```

0000  52 54 00 12 35 02 08 00  27 6d ac 0d 08 06 00 01  RT..5... 'm.....
0010  08 00 06 04 00 01 08 00  27 6d ac 0d 0a 00 02 0f  ..... 'm.....
0020  52 54 00 12 35 02 0a 00  02 02  RT..5... ..
  
```

pingCapture.pcapng | Packets: 10 · Displayed: 10 (100.0%) | Profile: Default

Type here to search

ENG 06.46  
US 31/07/2019

Figure 3: Capture Results

You can see we intercepted 10 packages in about 2 seconds.

Since my computer did not know how to get to cbs.dk, some address resolution was necessary. Now I want you to look at the protocol column. Some of these protocols should be similar from in class demoland activities.

ARP protocol to resolve IP addresses to MAC addresses (Layer 2).

DNS protocol to resolve URL (cbs.dk) to IP addresses.

Finally ICMP packages our ping command generated.

If you look at the source and destination addresses you will see that each packet we generated on our end (My IP address was 10.02.15) was answered by a response.

## DNS Packet

Let us take a look at the first DNS packet. Click the first DNS packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	PcsCompu_6d:ac:0d	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
2	0.000200	RealtekU_12:35:02	PcsCompu_6d:ac:0d	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
3	0.292677	10.0.2.15	10.0.2.3	DNS	66	Standard query 0xea09 A cbs.dk
4	0.294384	10.0.2.3	10.0.2.15	DNS	82	Standard query response 0xea09 A cbs.dk A 130.226.47.28
5	0.305552	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=15/3840, ttl=128 (reply in 6)
6	0.306352	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=15/3840, ttl=127 (request in 5)
7	1.343854	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=16/4096, ttl=128 (reply in 8)
8	1.344752	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=16/4096, ttl=127 (request in 7)
9	2.359190	10.0.2.15	130.226.47.28	ICMP	74	Echo (ping) request id=0x0001, seq=17/4352, ttl=128 (reply in 10)
10	2.360013	130.226.47.28	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0001, seq=17/4352, ttl=127 (request in 9)

> Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: PcsCompu_6d:ac:0d (08:00:27:6d:ac:0d), Dst: RealtekU_12:35:03 (52:54:00:12:35:03)
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.3
> User Datagram Protocol, Src Port: 56609, Dst Port: 53
> Domain Name System (query)

**Figure 4:** DNS 0

Now the middle part of the screen shows the information the packet contains.

> Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: PcsCompu_6d:ac:0d (08:00:27:6d:ac:0d), Dst: RealtekU_12:35:03 (52:54:00:12:35:03)
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.3
> User Datagram Protocol, Src Port: 56609, Dst Port: 53
> Domain Name System (query)

**Figure 5:** DNS 1

From top to bottom, these correspond to the five layers of the TCP/IP protocol stack.

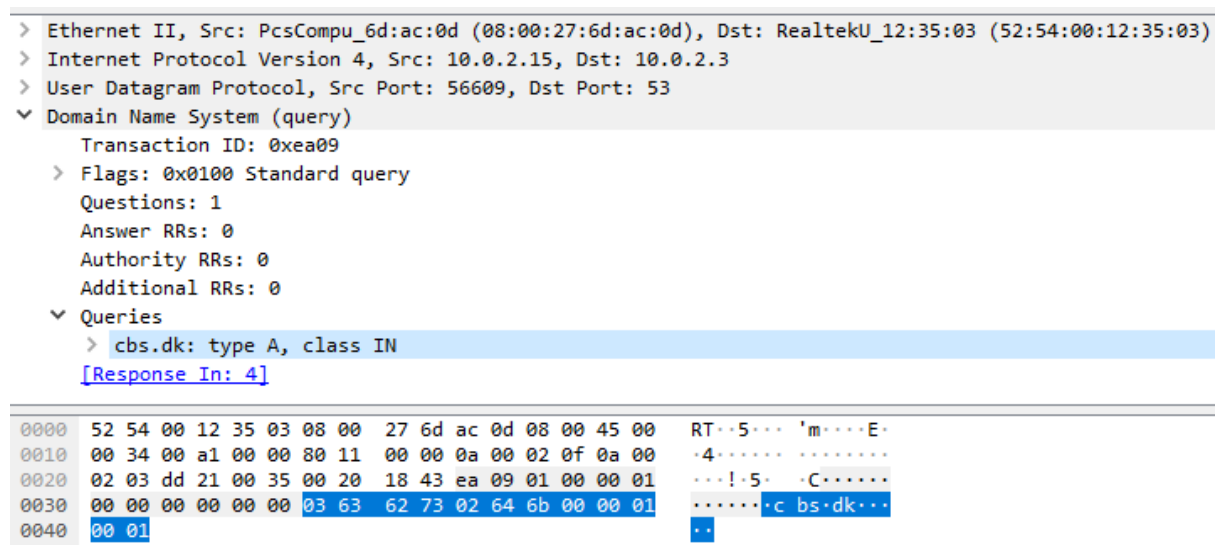
You can see the source and destination MAC addresses in layer 2 (Link).

Source and destination IP addresses in layer 3 (Network).

The transport protocol used (UDP) in layer 4 (Transport).

And finally the application protocol in layer 5 (DNS).

Let us investigate layer 5. We know that our computer asks the DNS server about cbs.dk. Click “Domain Name System (query)” to see the information contained in that layer.



**Figure 6:** DNS 2

You can see the interpretation of the protocol in human readable format in the middle pane. The bottom pane displays the hexadecimal representation of the data.

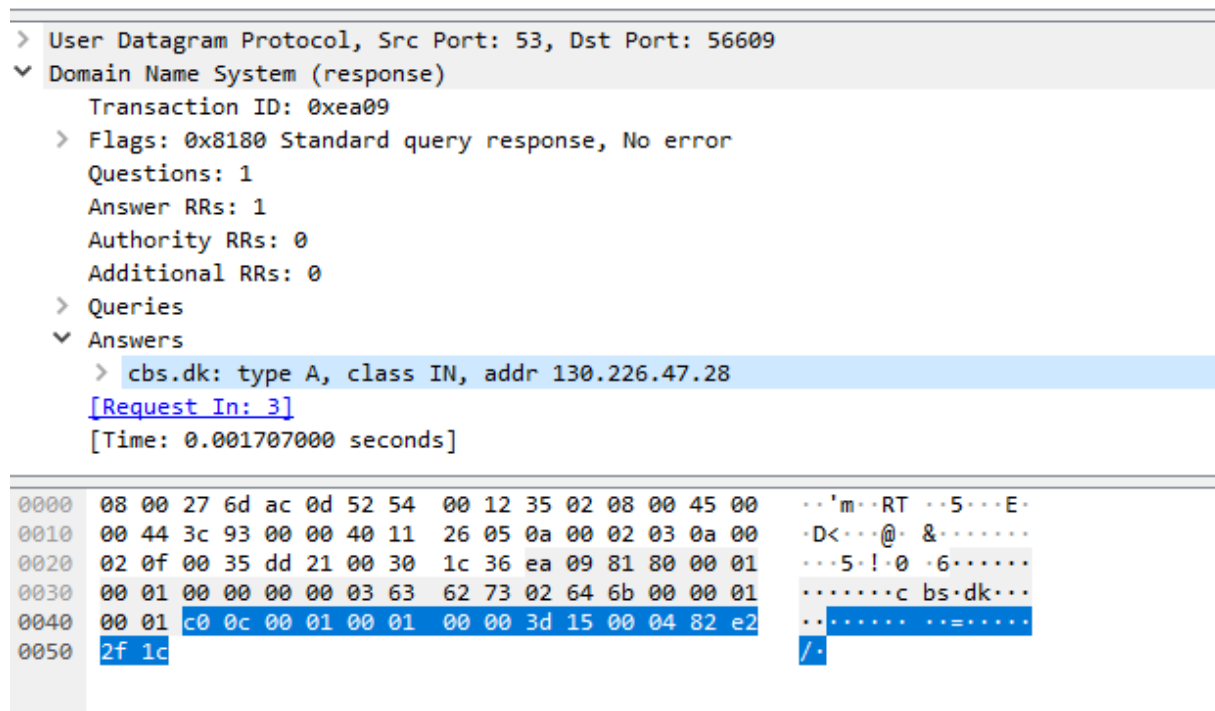
To finish up, let us inspect the response from the DNS server. Click on the next DNS packet.

You can see in the response the addresses of the destination and source are switched. So this message is from the DNS server (10.0.2.3) to my computer (10.0.2.15).

The answer indicates cbs.dk can be reached at 130.226.47.28.

## ICMP Ping

ICMP is a protocol used by network devices to communicate with each other. It is used to send error messages, and operational information. As such it is not used to exchange data. We used ICMP protocol to verify that cbs.dk was up.



**Figure 7:** DNS 3

Since ICMP is a layer 3 protocol (remember routers operate at layer 3) you will see that the transport layer headers are missing.

What I want you to note here is the time differences between requests and responses. You might be familiar with ping as a measure of latency in online games. Essentially it is measuring how long it takes to get information from the server.

## Wireshark Capture

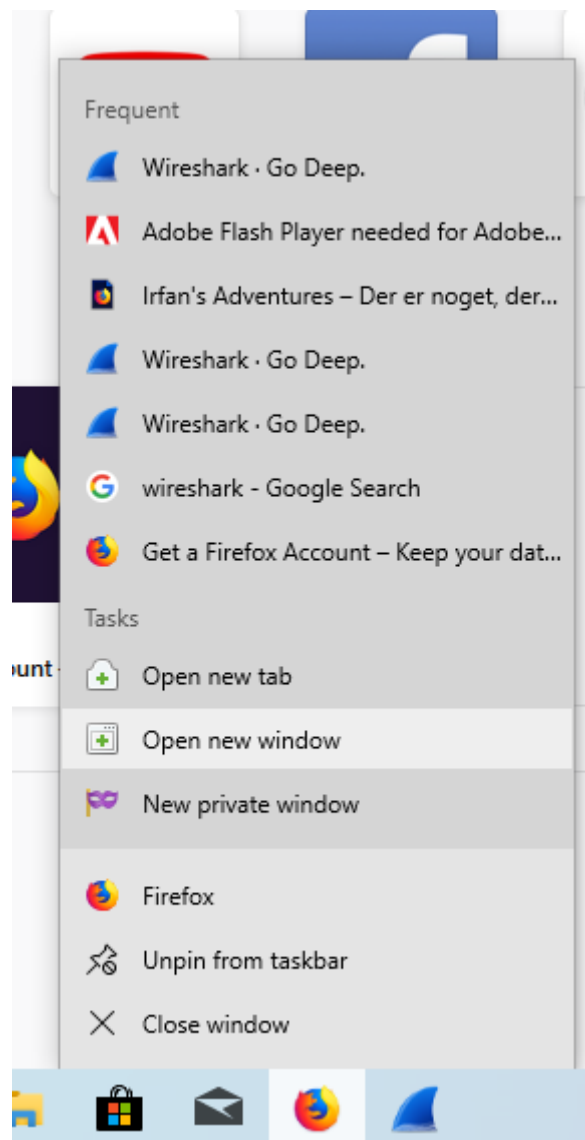
Now you know how to look at the data, you can also capture your own traffic. The capture you analyzed earlier was captured using Wireshark.

It is important to keep your network activity very limited during this exercise. Your computer generates quite a few packages by itself. You browsing the net during capture may produce an overwhelming number of packages. If you have services that use network data (dropbox, spotify, etc.), it may be a good idea to turn them off.

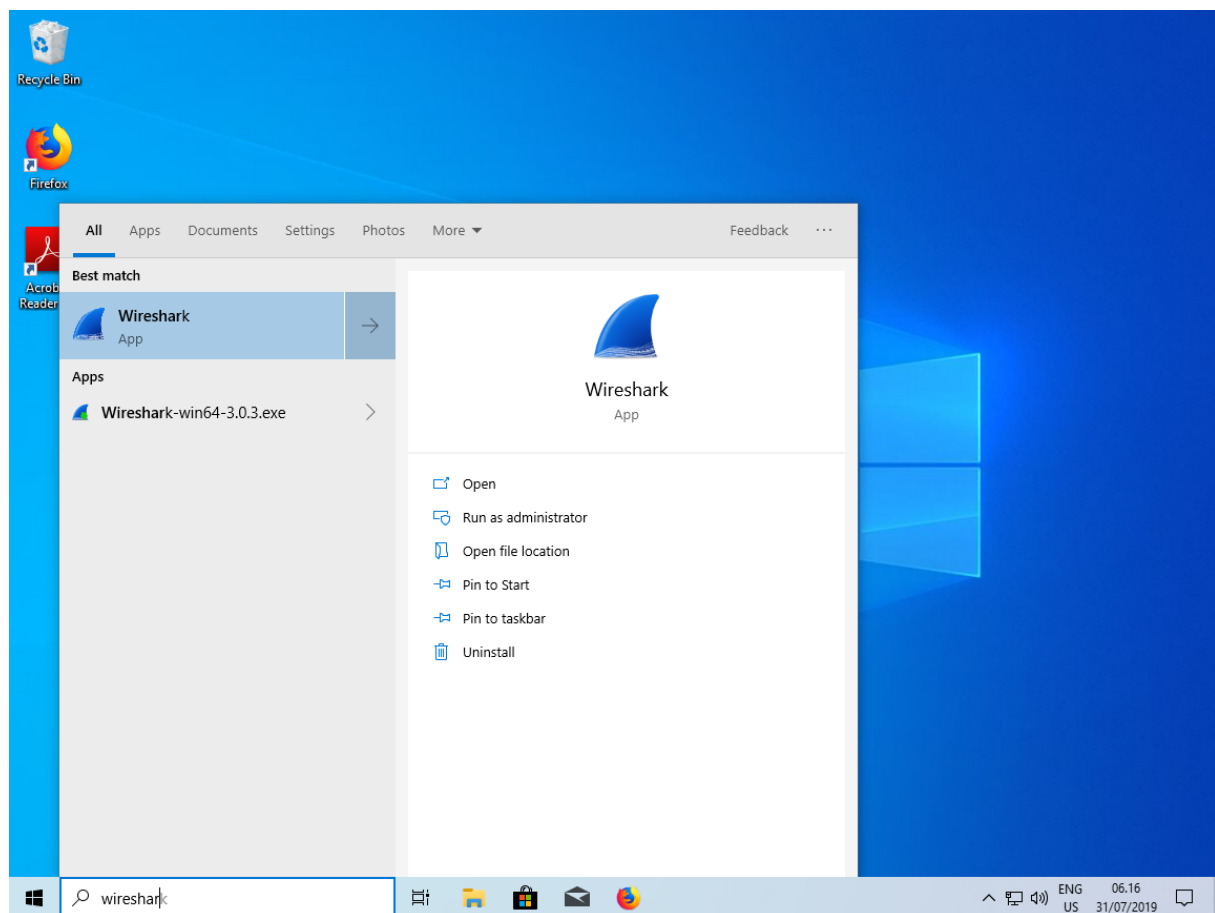
Open a new browser window without any tabs. Other tabs may create unnecessary traffic.

Launch wireshark. You can find it in the Start Menu.





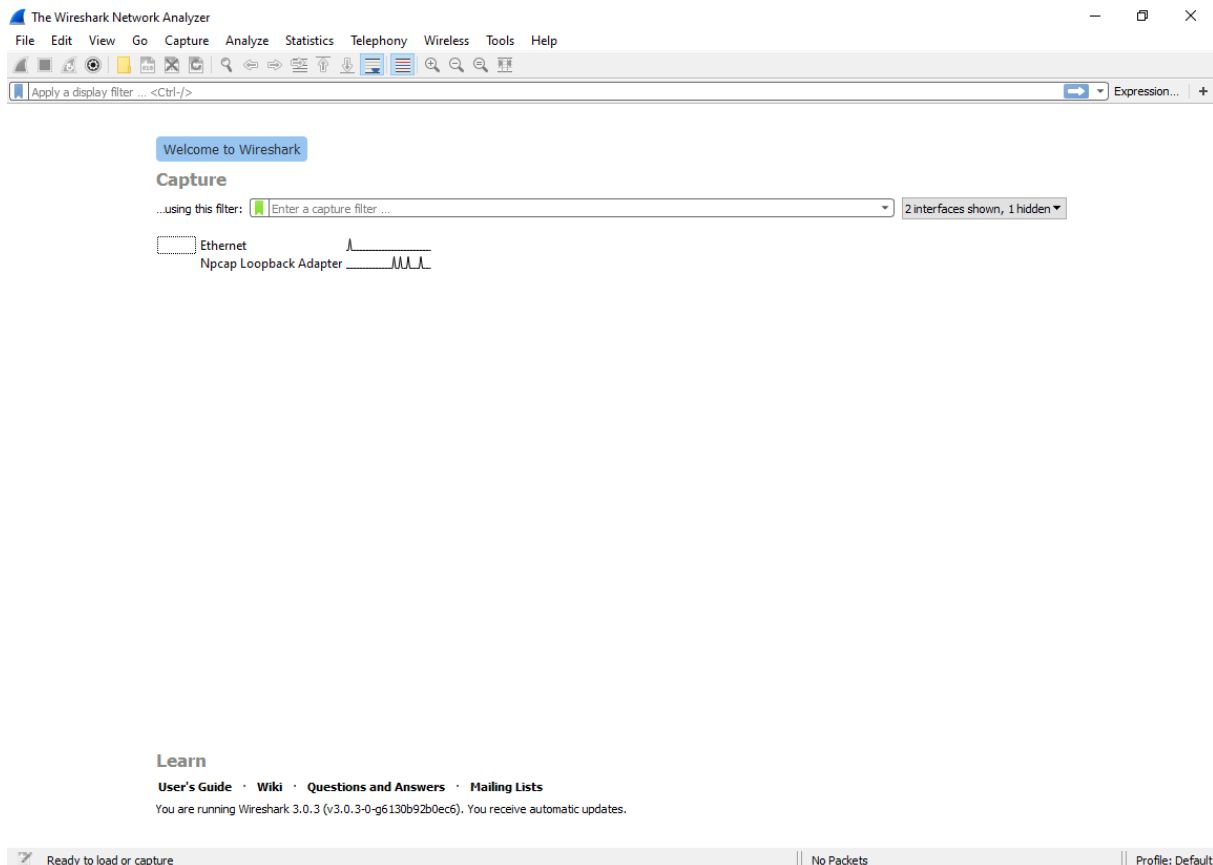
**Figure 8:** New Browser Window



**Figure 9:** Launch Wireshark

After wireshark initializes you can see a list of network interfaces.

Depending on your hardware, there will probably be a wired ethernet and a wireless ethernet adapter (and may be a few other odds and ends).



**Figure 10:** First Screen

Select the interface you want to use. Correct interface depends on how you connect to the network. I will be using wired ethernet (as my workstation does not have a wireless interface).

Once you select the interface the capture icon on the top left will become clickable. The capture icon looks like a shark fin.

After you click the capture button, switch to the browser window and type in cbs.dk in the address bar and press enter.

As soon as you see the cbs.dk landing page, switch back to wireshark and click the red square on the top right corner to stop capture. We don't want to capture any other traffic (It can get confusing otherwise).

This is how it should look like after the capture.

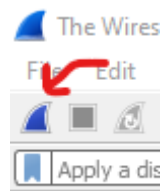


Figure 11: capture

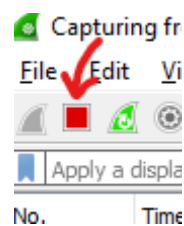


Figure 12: Stop Capture

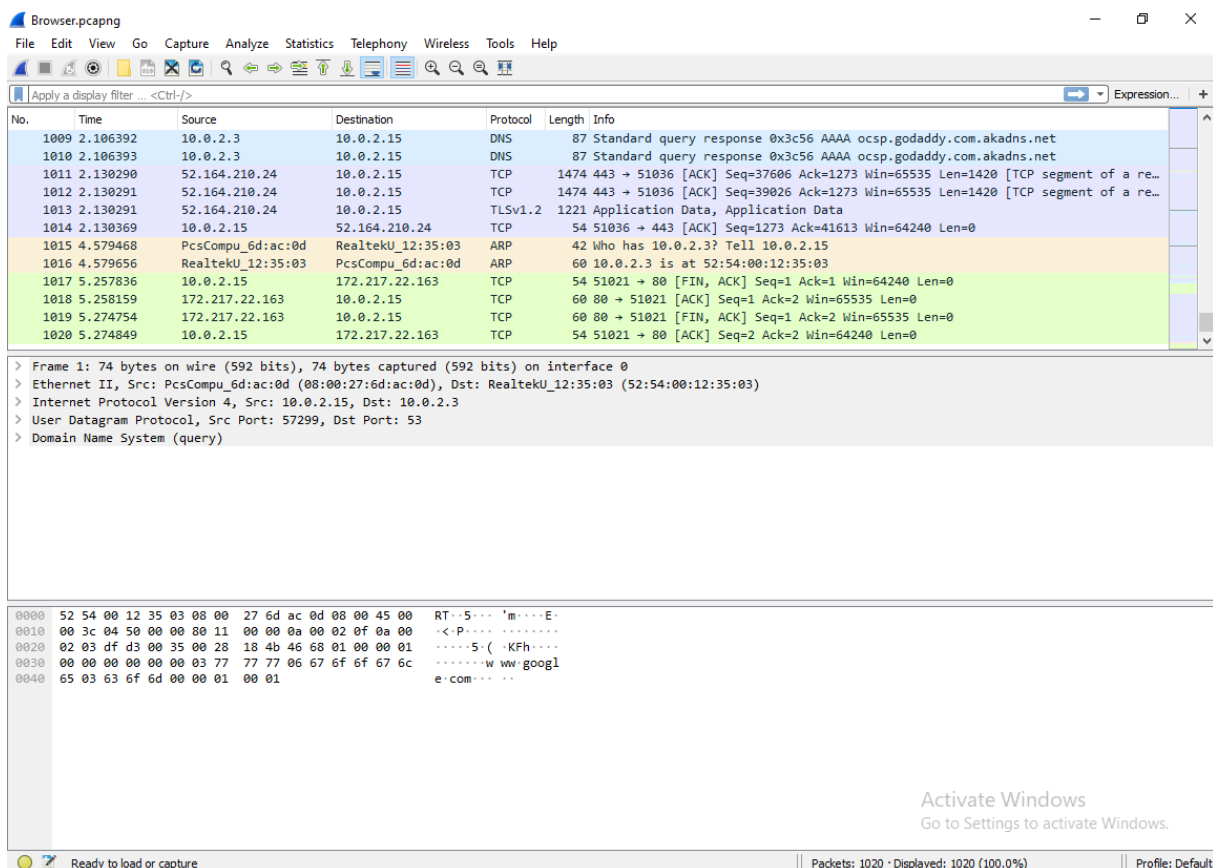


Figure 13: Browser Capture

## Wireshark Analysis - Filters

That visit to CBS.dk generated over a thousand packages in the 5 seconds it took me to complete it. For compatibility purposes, find attached my capture.

### Browser Capture

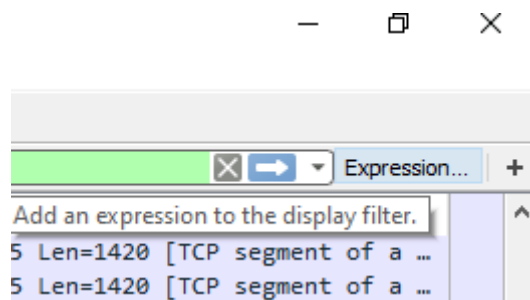
Making sense of ten packets as we did before is easy. When it is real traffic, we need some filtering capabilities. The true beauty of Wireshark is the filtering capabilities.

Let us start small.

1 - As you saw, there is a lot of traffic. Let's say I am interested only in packages to and from cbs.dk.

We know that the IP address of cbs.dk is 130.226.47.28. We can create a filter to show only the packages that contain 130.226.47.28 as the IP address.

Click the expression next to the filter bar.



**Figure 14:** Expression

Now you see a long list of protocols that are used in modern networking. We talked about 5 layers. These are the protocols that populate those layers.

So the protocol we are interested in is the IP protocol. Scroll down the list until you see IPv4.

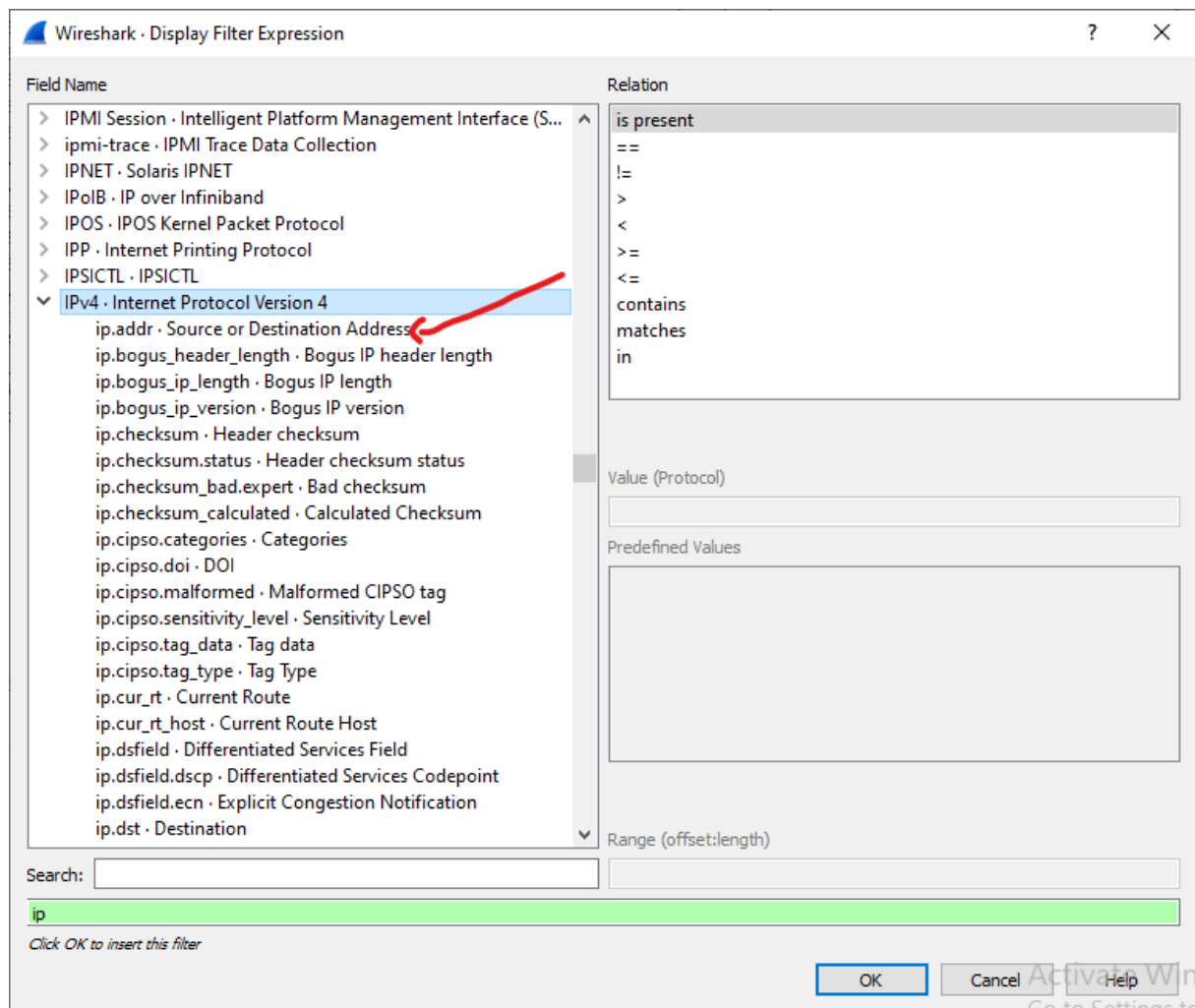
We need ip.addr as the field, == as the relation, and 130.226.47.28 as the value.

2 - It is also possible to link multiple filters together with logical operators (and, or, not...).

For example let us say I am only interested in encrypted traffic using TLS protocol. We can change the filter to read:

```
ip.addr == 130.226.47.28 and tls
```

3 - Now we know the packages related to the web content coming from cbs.dk. What about all the rest?

**Figure 15:** filter

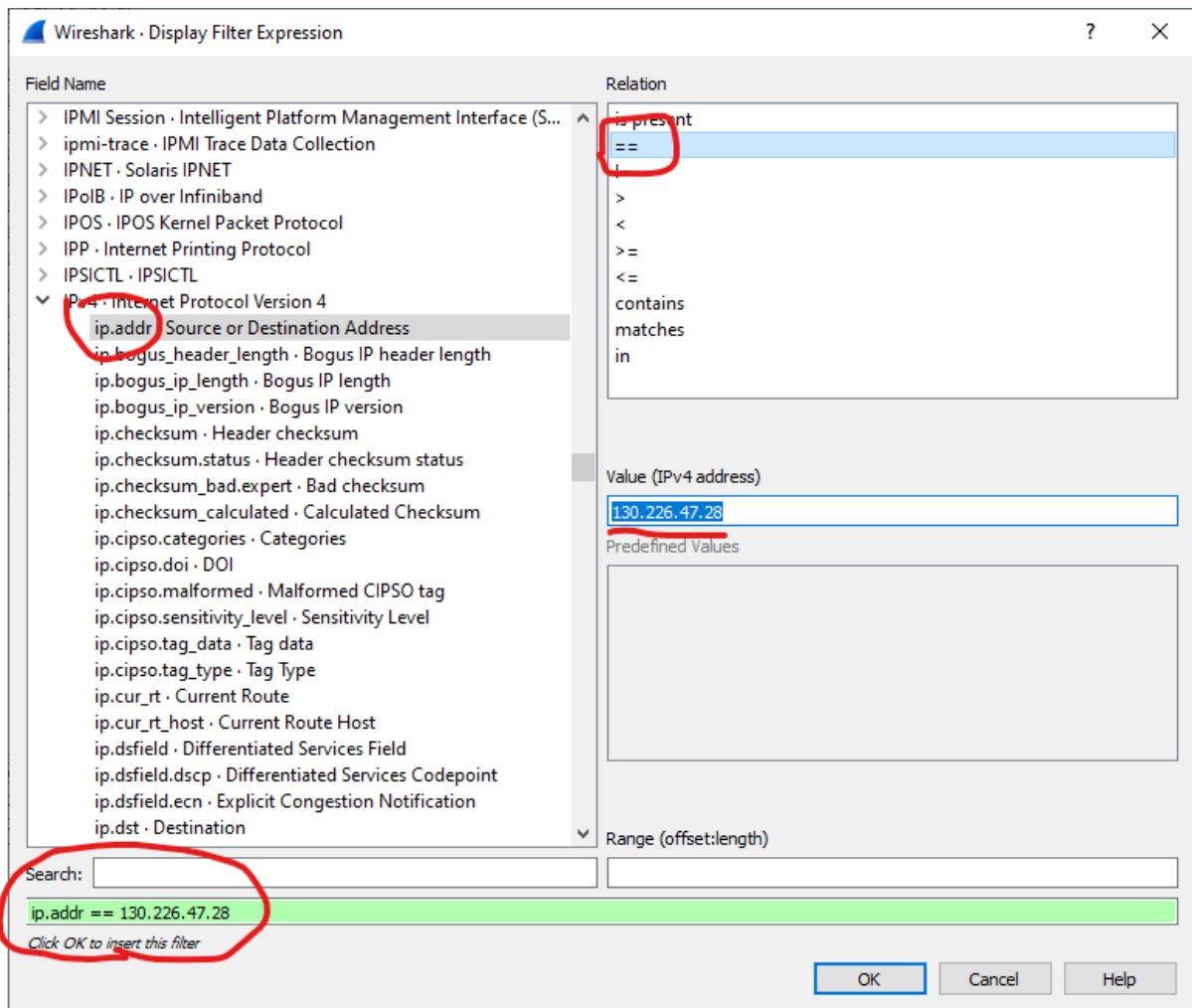
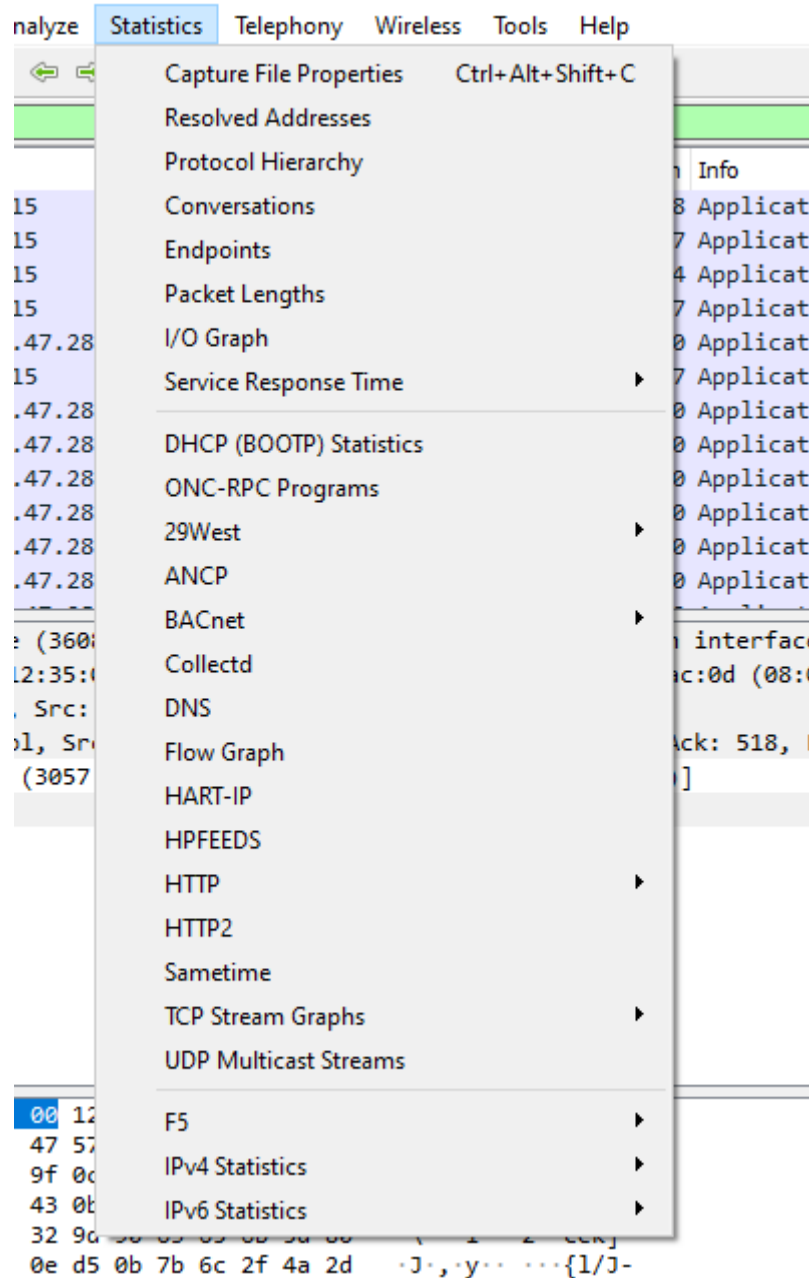


Figure 16: filter 2

File Edit View Go Capture Analyze Statistics Te				
ip.addr == 130.226.47.28 and tls				
No.	Time	Source	Desti	
112	1.255557	10.0.2.15	130.	
114	1.256518	130.226.47.28	10.0	
115	1.256643	130.226.47.28	10.0	
119	1.257318	130.226.47.28	10.0	
121	1.257692	130.226.47.28	10.0	
131	1.296514	10.0.2.15	130.	
133	1.296977	10.0.2.15	130.	

Figure 17: Logical Filter

**Figure 18:** Statistics



Let us look at some statistics.

Statistics > Choose Capture File Properties.

#### Interfaces

<u>Interface</u>	<u>Dropped packets</u>	<u>Capture filter</u>	<u>Link type</u>	<u>Packet size limit</u>
Ethernet	Unknown	none	Ethernet	262144 bytes

#### Statistics

<u>Measurement</u>	<u>Captured</u>	<u>Displayed</u>	<u>Marked</u>
Packets	1020	170 (16.7%)	—
Time span, s	5.275	0.600	—
Average pps	193.4	283.4	—
Average packet size, B	723	715	—
Bytes	737163	121517 (16.5%)	0
Average bytes/s	139 k	202 k	—
Average bits/s	1118 k	1620 k	—

**Figure 19:** stats

The packages we filtered out as content make up only 17% of the packages. You can broaden the filter by removing tls and notice that the traffic from cbs.dk makes up only 60% of the packages captured.

4 - Where are all these other packages coming from?

Statistics > Resolved Addresses

17 different IP addresses. Mostly related to advertisements.

5 - Investigate the ports associated with IP destinations to see what services are being used.

Statistics > IPv4 Statistics > Destinations and Ports

Remember ports are used to direct traffic to programs installed on a computer. Some of these ports are registered for certain purposes.

Port 80 is used by web servers for HTTP traffic.

Port 443 is used for encrypted version of HTTP.

You will see other ports as well. You can look up what they are on google.

## Wireshark Conclusion

There is so much more you can do with wireshark. Using wireshark you can detect intruders in your home wifi network (Look at MAC addresses), or bitcoin mining trojans installed on your computer (anyone using bitcoin protocol?).

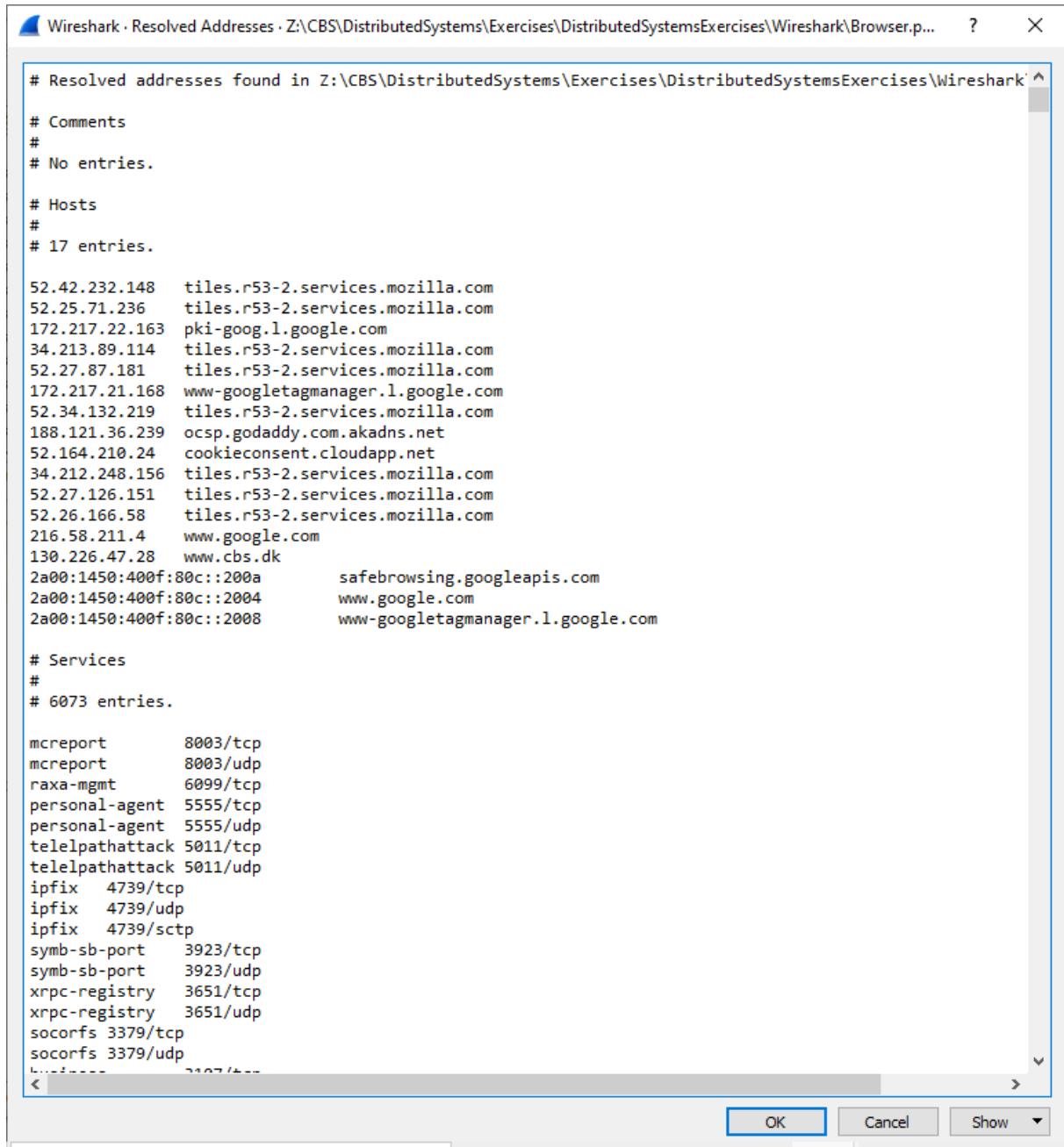
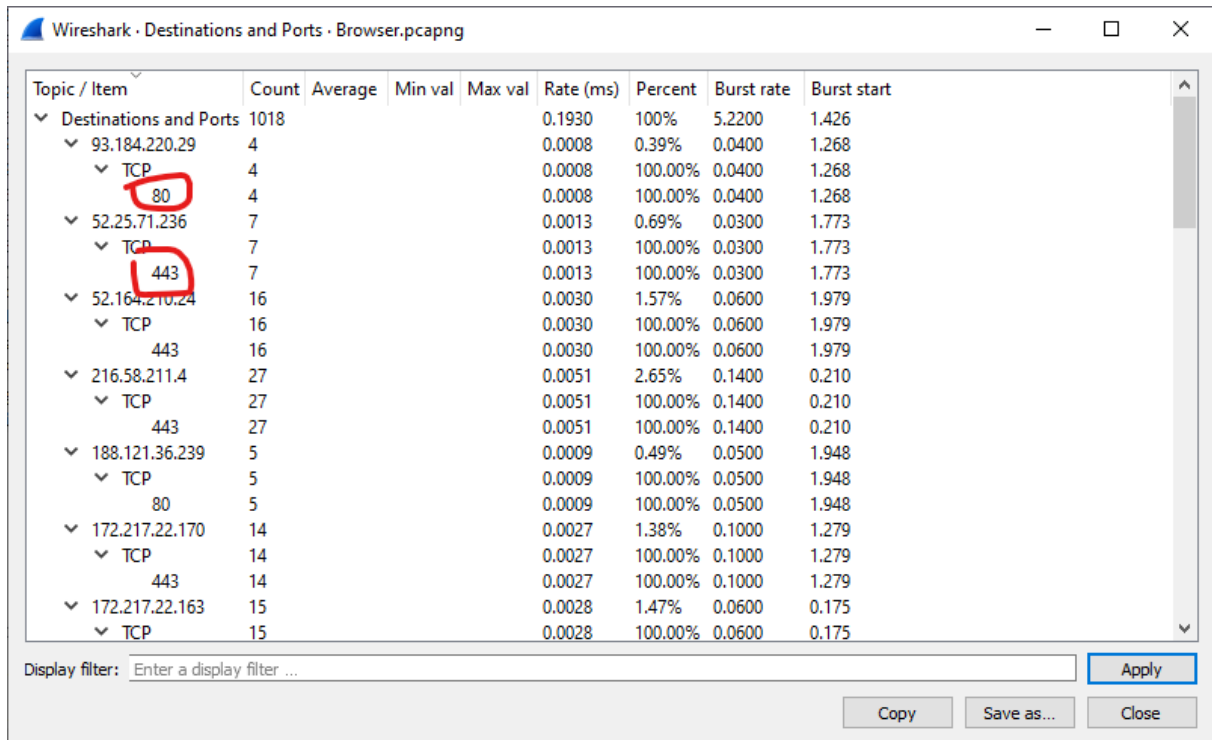


Figure 20: hosts



Wireshark · Destinations and Ports · Browser.pcapng

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Destinations and Ports	1018				0.1930	100%	5.2200	1.426
▼ 93.184.220.29	4				0.0008	0.39%	0.0400	1.268
▼ TCP	4				0.0008	100.00%	0.0400	1.268
80	4				0.0008	100.00%	0.0400	1.268
▼ 52.25.71.236	7				0.0013	0.69%	0.0300	1.773
▼ TCP	7				0.0013	100.00%	0.0300	1.773
443	7				0.0013	100.00%	0.0300	1.773
▼ 52.164.210.24	16				0.0030	1.57%	0.0600	1.979
▼ TCP	16				0.0030	100.00%	0.0600	1.979
443	16				0.0030	100.00%	0.0600	1.979
▼ 216.58.211.4	27				0.0051	2.65%	0.1400	0.210
▼ TCP	27				0.0051	100.00%	0.1400	0.210
443	27				0.0051	100.00%	0.1400	0.210
▼ 188.121.36.239	5				0.0009	0.49%	0.0500	1.948
▼ TCP	5				0.0009	100.00%	0.0500	1.948
80	5				0.0009	100.00%	0.0500	1.948
▼ 172.217.22.170	14				0.0027	1.38%	0.1000	1.279
▼ TCP	14				0.0027	100.00%	0.1000	1.279
443	14				0.0027	100.00%	0.1000	1.279
▼ 172.217.22.163	15				0.0028	1.47%	0.0600	0.175
▼ TCP	15				0.0028	100.00%	0.0600	0.175

Display filter:

**Figure 21:** Ports

What we did today is just scratching the surface, but if you are interested you can add this as a great tool under your tool belt.