Thomas Murphy
Cyb 605-Z3 – Principles of Cybersecurity
Wireshark Lab
Oct 3, 2016

Introduction

This lab introduces packet sniffing and packet analysis; the process of capturing any data passed over the local network and looking for any information that may be useful. Packet sniffers can be hardware appliances or software based (Hannah, 2011). A sniffer works by placing the network interface you want to listen to into promiscuous mode, thus reporting all the packets that it sees (Weadock, 2009).

The lab specifically focuses on the use of Wireshark, a graphical packet sniffer, and packet sniffing for security professionals. Wireshark (and tcpdump) enable administrators to view and examine packets in a granular format in either real-time or to a capture file for examination later. This provides a tool for the administrator to find network errors, measure bandwidth, become aware of intrusion attempts and attacks, map and discover network devices, and become aware of protocols and applications being used on the network.

In this lab I'm using CentOS version 7.2 and A VMWare version of Windows 95. I've run Wireshark over my wireless home network with approximately 20 nodes, including cell phones, two Roku devices, a Brother printer , various laptops, and baby monitors. I've also run it on a corporate network (with permission) that contains high sped market data and multicast networking. Lastly I've also captured data at a public wireless hotspot that required no passwords.

Objectives

The objectives of this lab are to execute the Wireshark Lab assignments in order to review and enhance previous learning, as well as to learn through experience, and gain the skills and knowledge about various tools currently available. In order to achieve the objective I've connected to three networks using my Dell Laptop running Centos 7.2 through its wireless network interface. I captured packets over a period of time, both for live capture into capture files for later analysis. I applied various filtering to isolate specific types of network packets and described my finding in detail.

Definitions

- Wireshark
    a) **Sniffer** - software that intercepts and logs network traffic that it can "see" on a network interface. It's used for capturing and reporting data flows. (Paessler, 1996)

    b) **Protocol –** a set of agreed upon rules for how data is transmitted. In terms of network protocols, there are protocols for how data is packaged and exchanged across a network. The most common of these protocols is "TCP/IP". There are also application protocols that define how a client and server will communicate upon making a network connection (Hunt 1992),

    c) **Protocol Analyzer –** A tool used to analyze protocol interaction between two systems (Hunt, 1992)

    d) **Packet –** Data that is communicated across a network is broken into small blocks called packets. "Dividing the data into packets helps the sender and receiver of the data decide

which packets arrived in tact". "Dividing data into packets also ensures that multiple network connections can share a network device" (Comer, 1997)

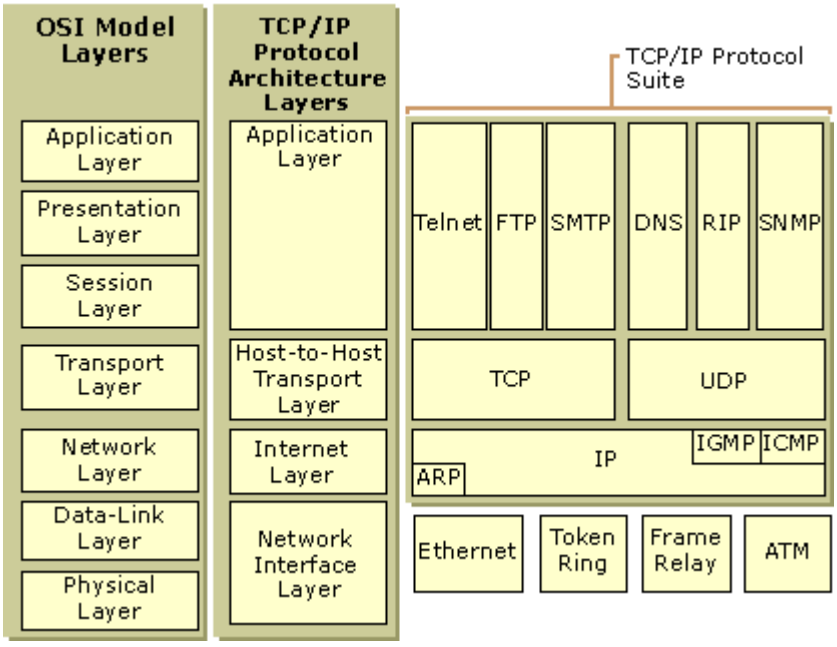
e) **Packet Filter –** A tool used for selectively accepting or blocking data as it passes through a network interface (OpenBSD, 2016)

f) **Live Capture –** there are two ways to capture data; live and into a file for analysis later on.

- TCP/IP
  a) **Transport Control Protocol (TCP)** - A connection oriented protocol that provides a full-duplex byte stream for user processes. This means, that when a packet is sent to a host, an acknowledgement packet is returned to the sender. (Stevens, 1990)

  b) **Internet Protocol (IP) –** This is generally combined with TCP for the purpose of routing and delivering the TCP packet (Stevens, 1990)

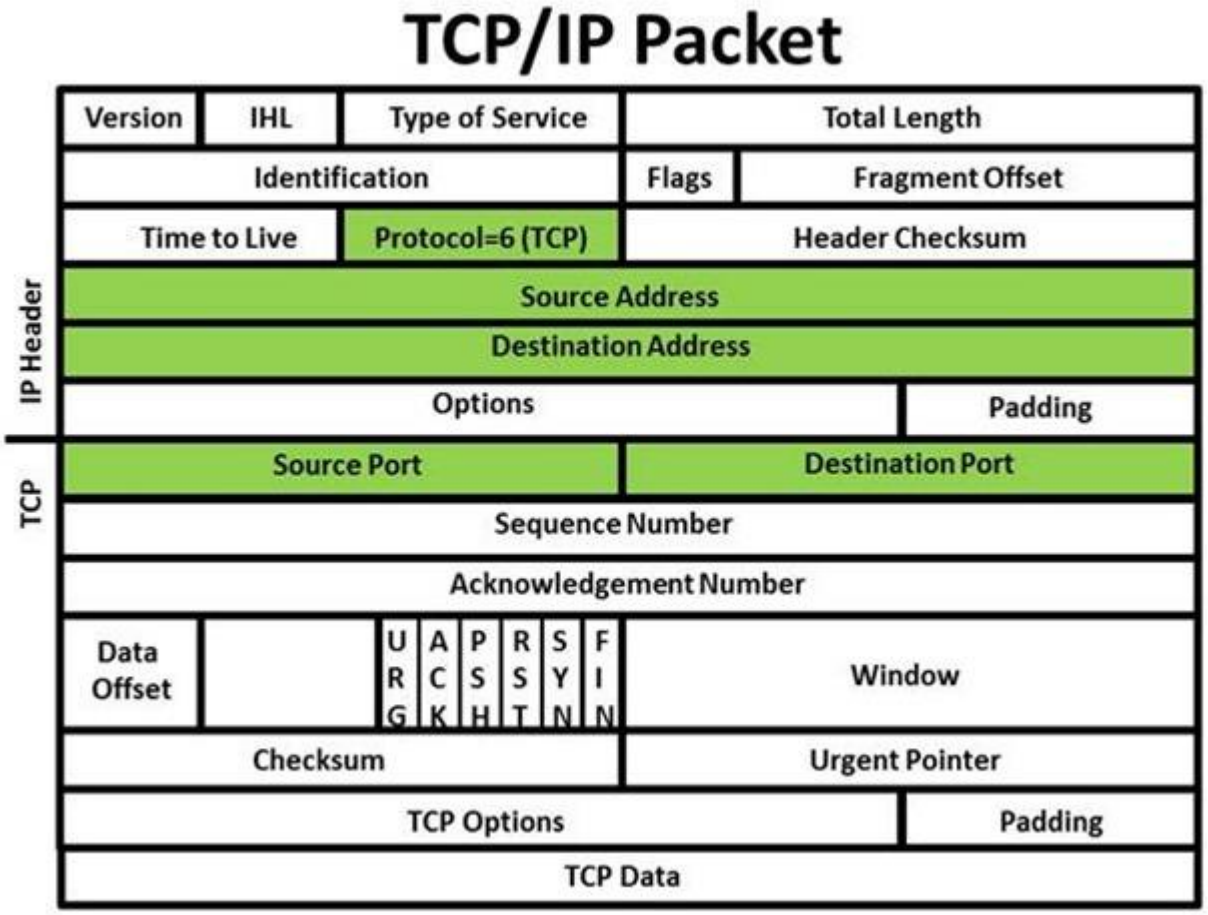
  c) **Network Stack –** A network stack is a reference to the layers of protocols necessary for data communication. When discussing a network stack we generally refer to the OSI model of network communications.



The OSI model is a theoretical model of communication. In the above diagram, you can see how the TCP/IP Network Stack lines up against OSI and shows examples of each layer in the suite. (Erbay, 2016)

d) **Packet Header –** A packet contains two parts, a header and a payload. The header contains information about where the packet should be sent and details about the contents of the packet, while the payload contains the data the is actually being sent. A typical IPV4 packet contains 20 bytes of data. The 20 bytes of data contain the IP version (4 for IPv4), IP Header Length, type of service, datagram length, an ID tag to

help reconstruct, instructions on fragmentation, fragmentation offset (what part of the packets to start reconstruction), TTL (number of network hops), a header checksum (to help detect errors), source IP address and the destination IP address (Comer, 1997).



Using Wireshark

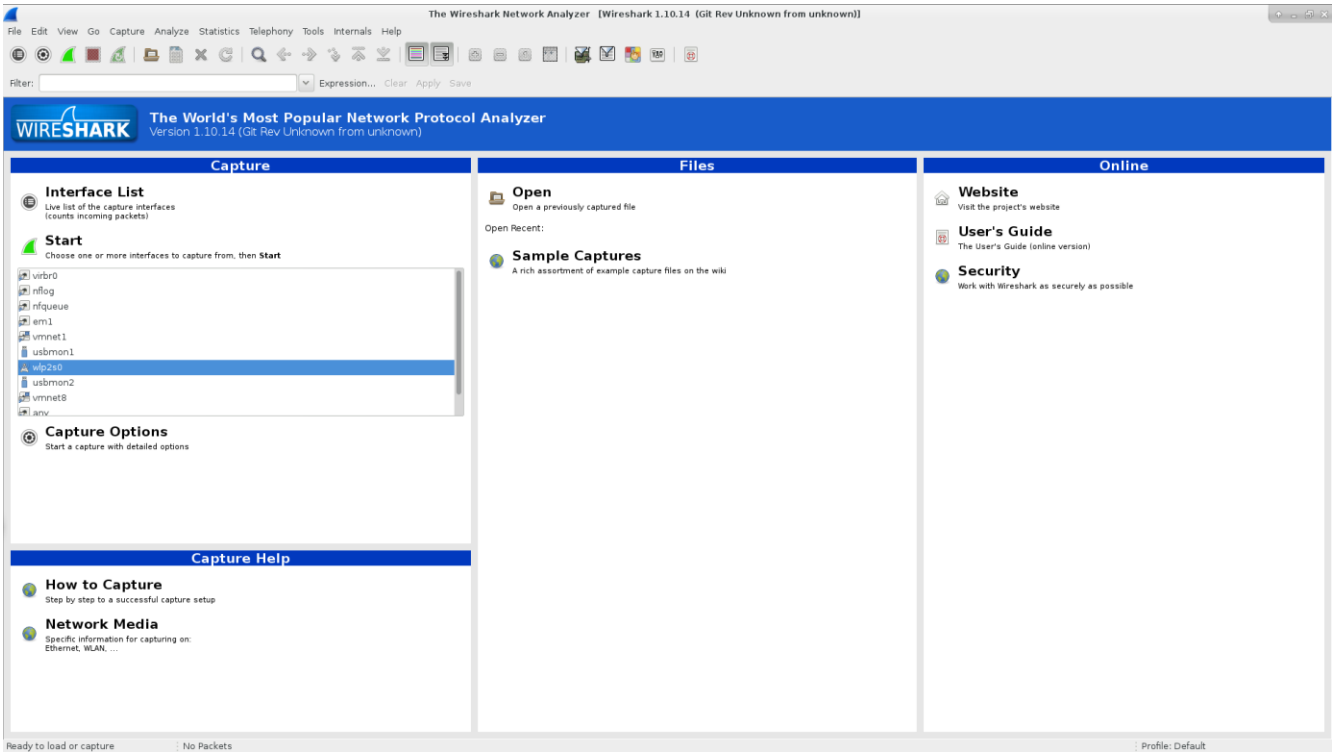
This section was an introduction for starting Wireshark, finding documentation and beginning to sniff data from a network interface.

Procedure:

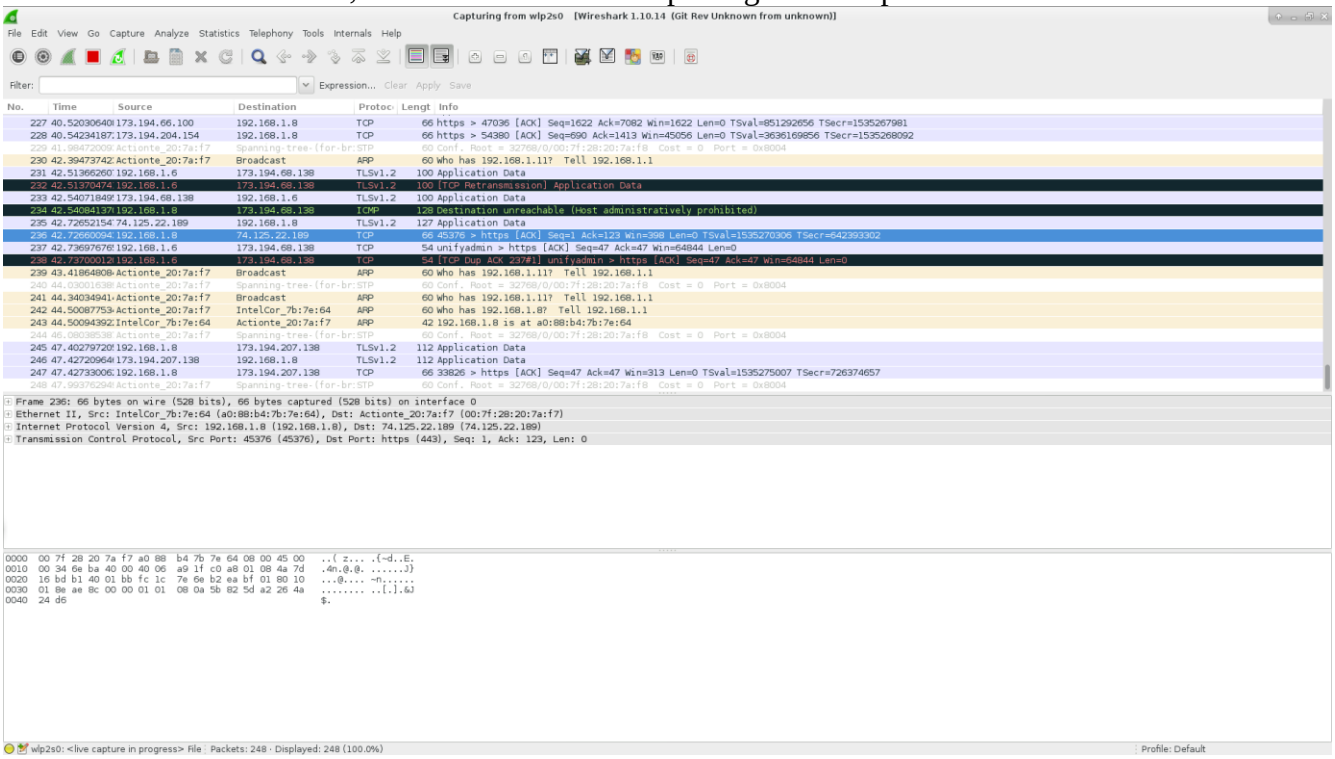
Steps 1 and 2 of the lab are for downloading and installing Wireshark and all other necessary packages. For Wireshark on Centos 7.2 (or any Red Hat Enterprise Linux variant) the command is simply:

```
yum –y install wireshark*
```

Step 3: Upon starting Wireshark we were asked to select our Internet facing interface



Once the NIC was selected, we were able to start capturing network packets:

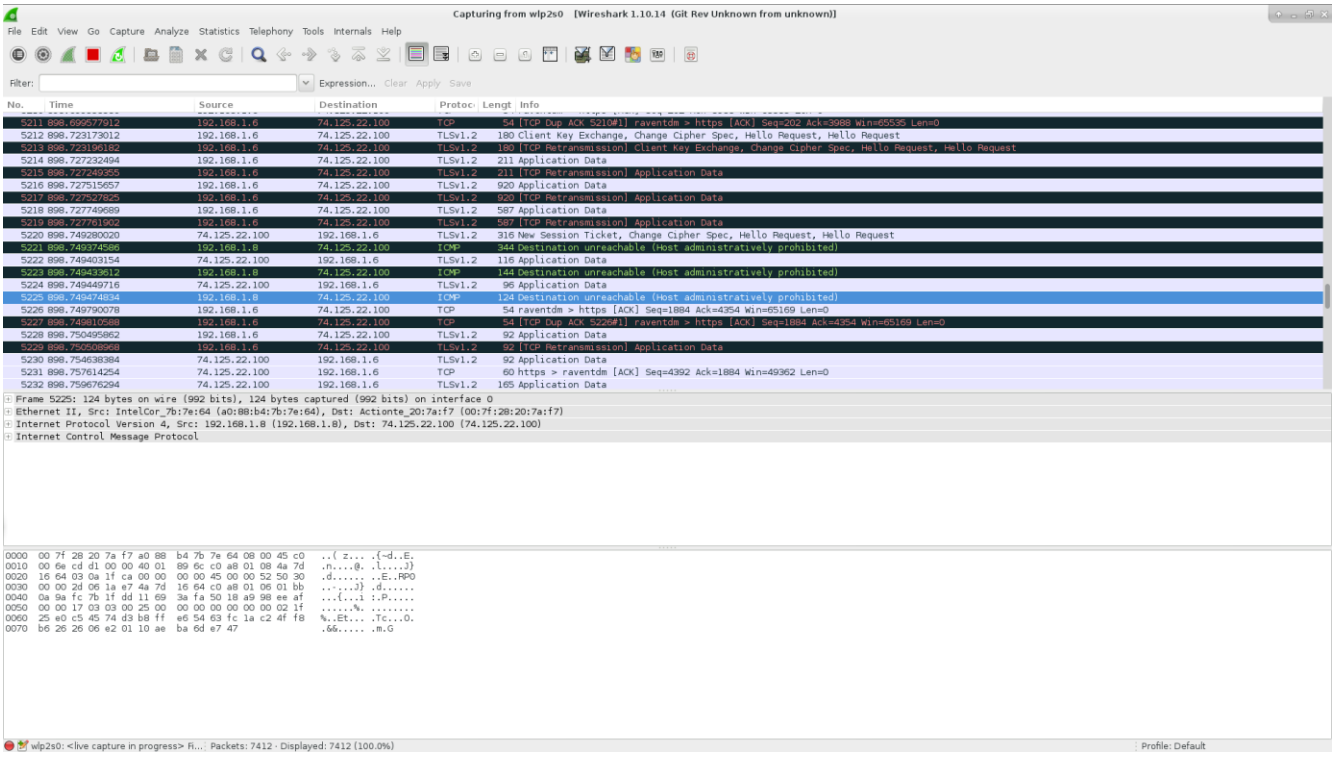


From the picture above you can note that the GUI is broken up into three main panels.

1. **The packet frame -** This displays general contents in the capture file including, the
   - Number off packets.
   - Time - The timestamp of the packet.
   - Source - The address where this packet is coming from.
   - Destination - The address where this packet is going to.
   - Protocol The protocol name in a short version.
   - Length The length of each packet.
   - Additional information about the packet content. (Wireshark, 2014)

2. **The "Packet Details" pane -** This shows the protocols and the protocol fields of the packet. Of special note, any data enclosed in "[" and "]" brackets is generated by Wireshark. Additionally, if Wireshark detects a relationship to another packet, it will create a link to that packet (Wireshark, 2014).

3. **The "Packet Bytes" pane -** This frame contains a hex dump of the entire packet (Wireshark, 2014)

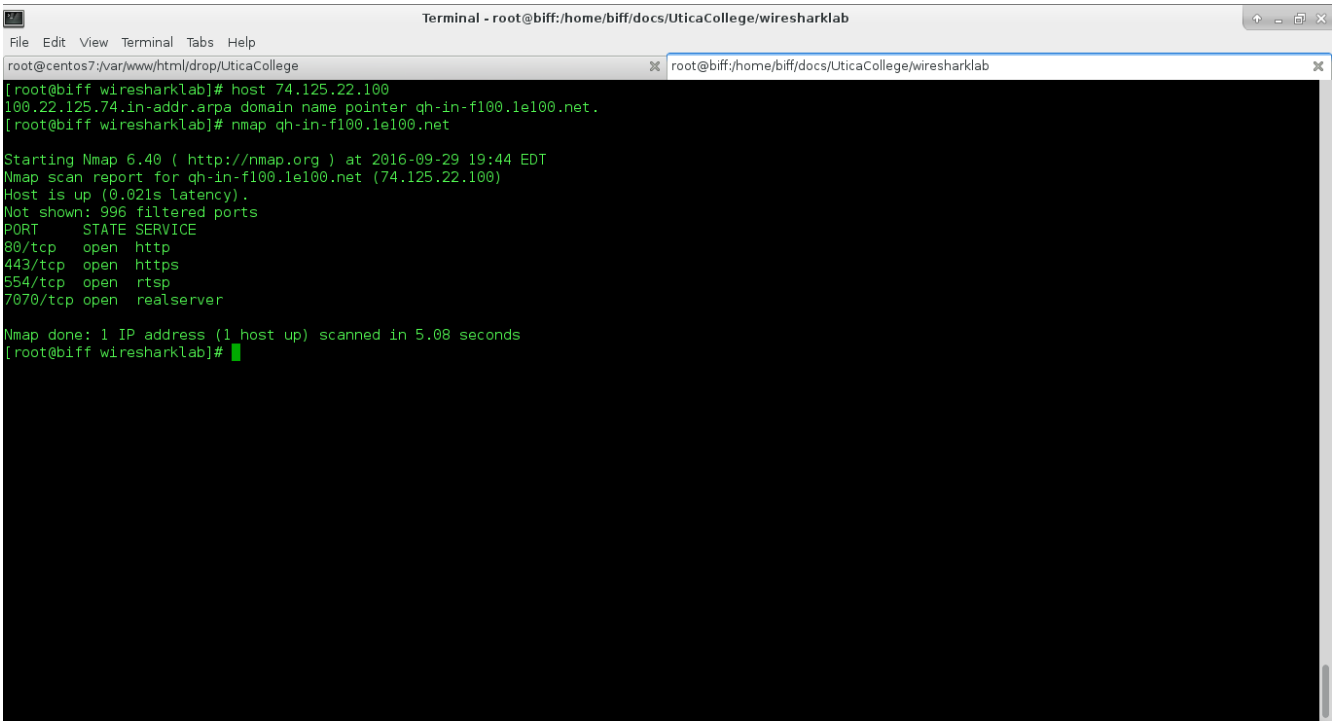


In my first run, I started seeing some communication with 74.125.22.100. I also noticed some retransmission packets and packets containing the following messages:

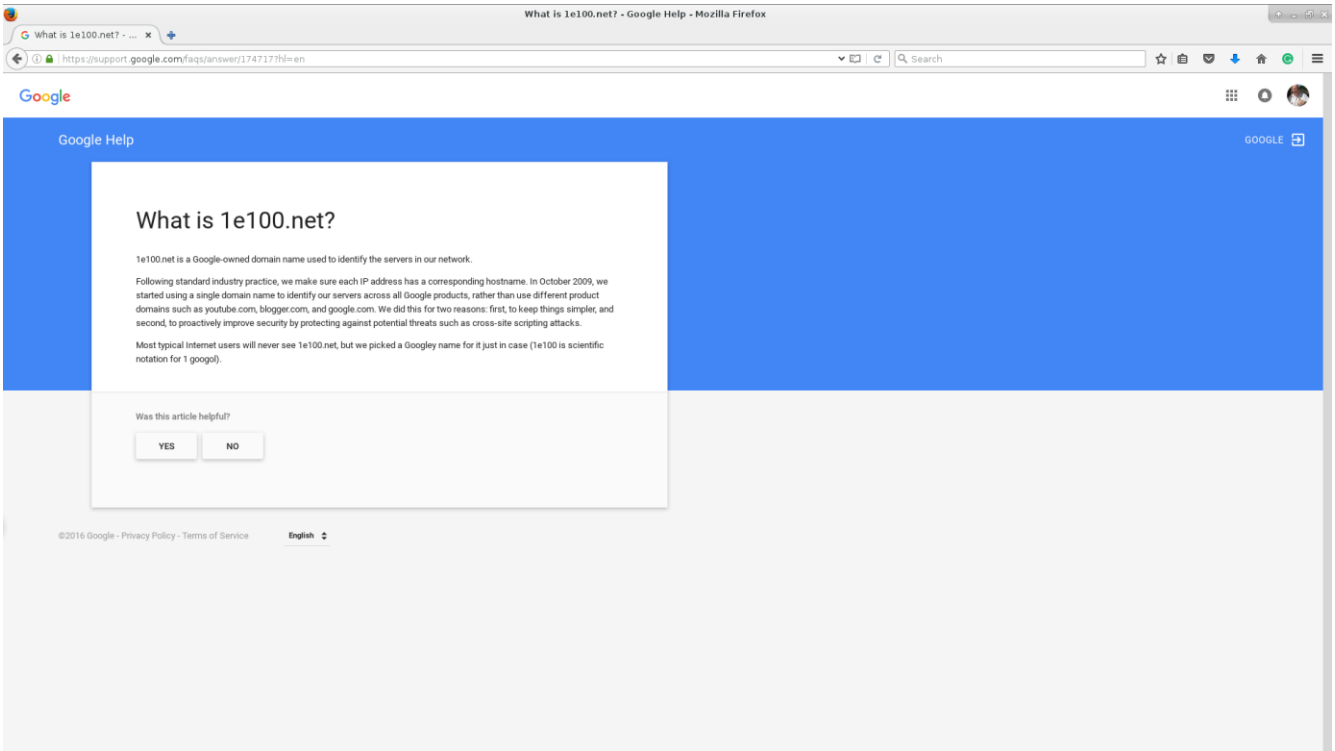
```
destination host is unreachable (host administratively prohibited)
```

They appeared to be only for ICMP packets, so I imagine they must be blocking those packets using a firewall or other network policy.

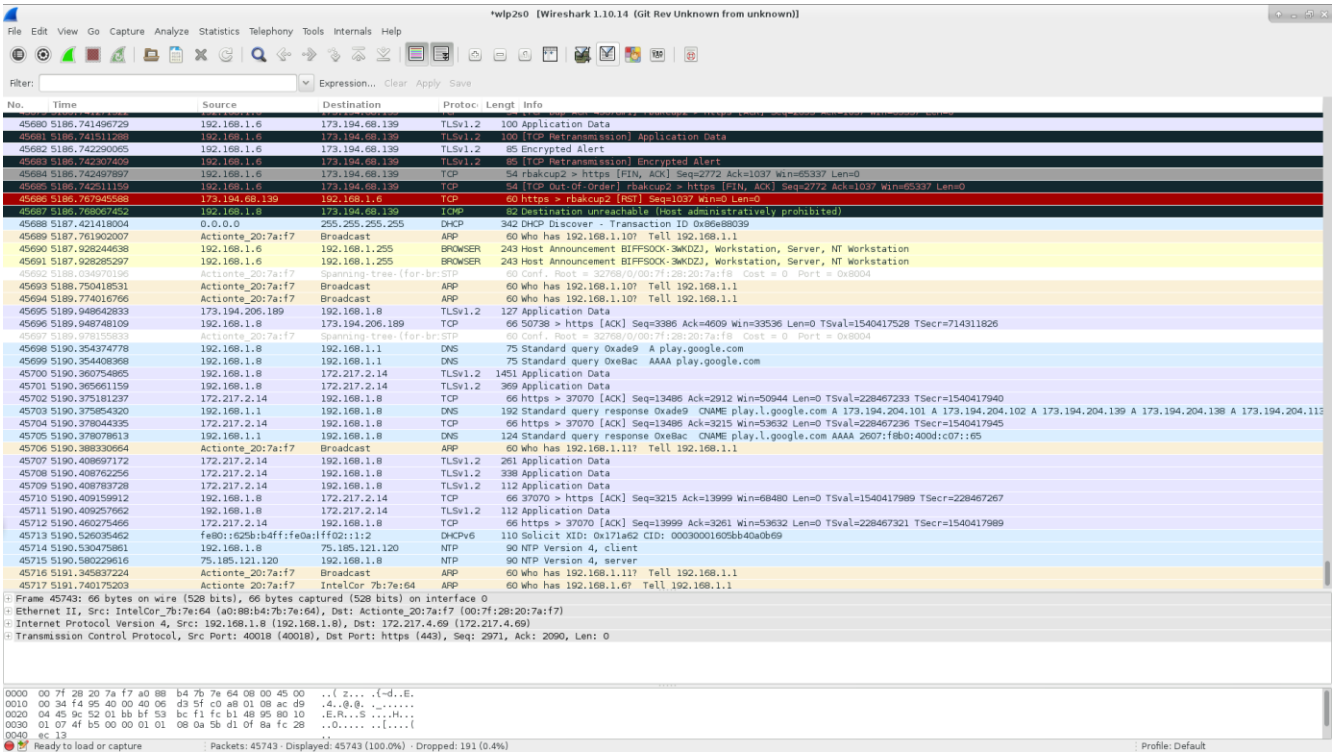
Some further investigation provided additional information:



It appears that this is Google:
https://support.google.com/faqs/answer/174717?hl=en

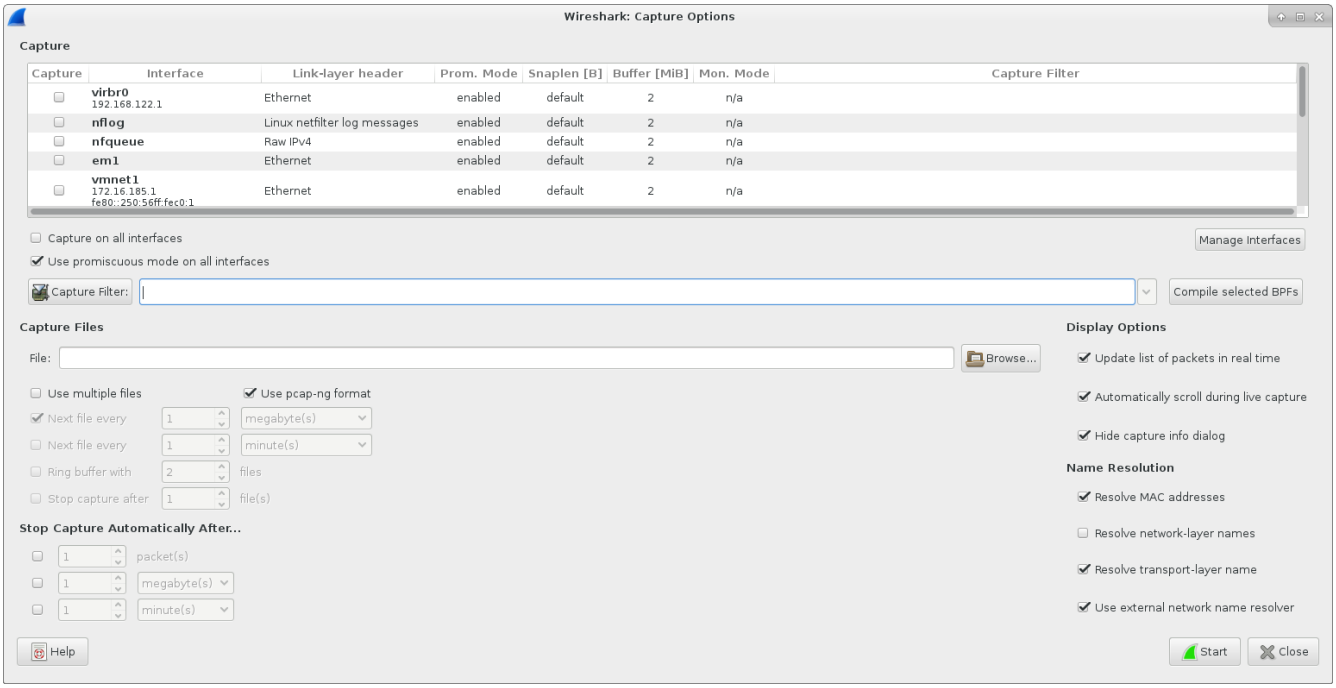


The lab then asks us to select the STOP button and describe the results in your lab:



I saw https, http, ssh, DNS, DHCP, ARP, TCP and NTP traffic. I also saw some duplicate packets, which according to Wireshark appear to be normal behavior. This will happen if the sending side transmitted a packet correctly, but thinks that it wasn't received at all (Wireshark, 2008).

We are then directed to select capture options; specifically “Update list of packets in real-time” and “Automatically scroll during live capture”. For CentOS 7.2, these were already the default options, so I made no changes.

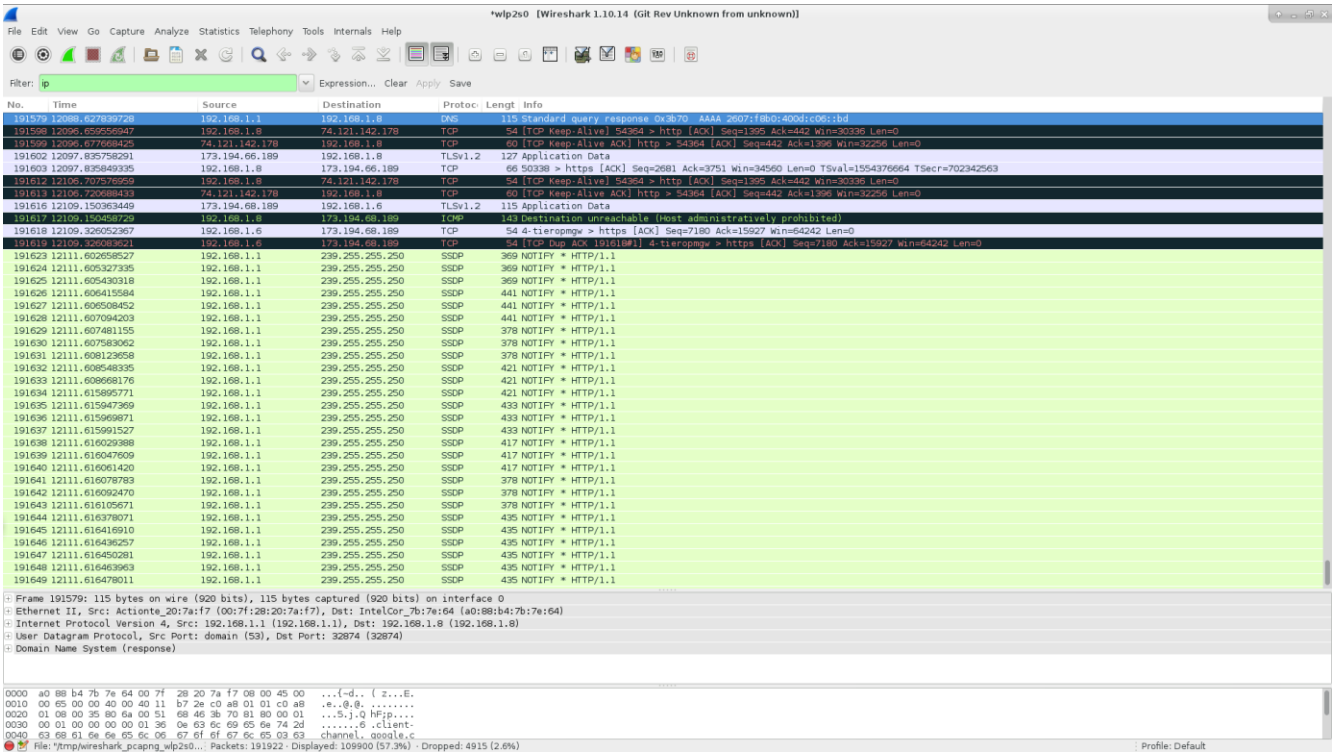


I played with turning on/off those options. The results were intended to show that packets were updated in the packet frame window in real-time (no buffering) as well as enabling scrolling using the scroll bar on the right side of the screen.

Filter Packets with the Filter Bar

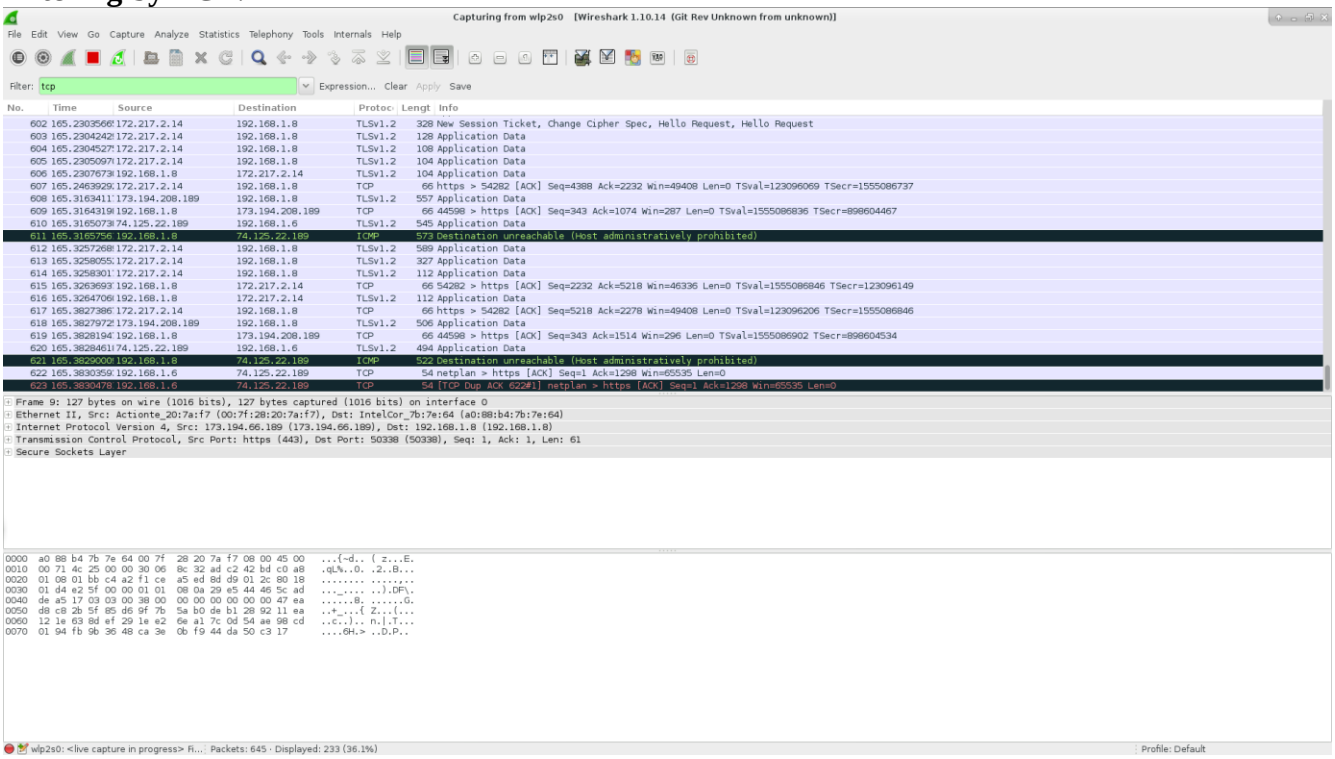
Our lab now asks us to filter packets. Without filtering, it's difficult to pinpoint what exactly we're looking for because of the glut of information. By narrowing down our search scope, we can more easily find specific things. The GUI contains a "Filter" bar that allows for expressions.

Filtering by IP:



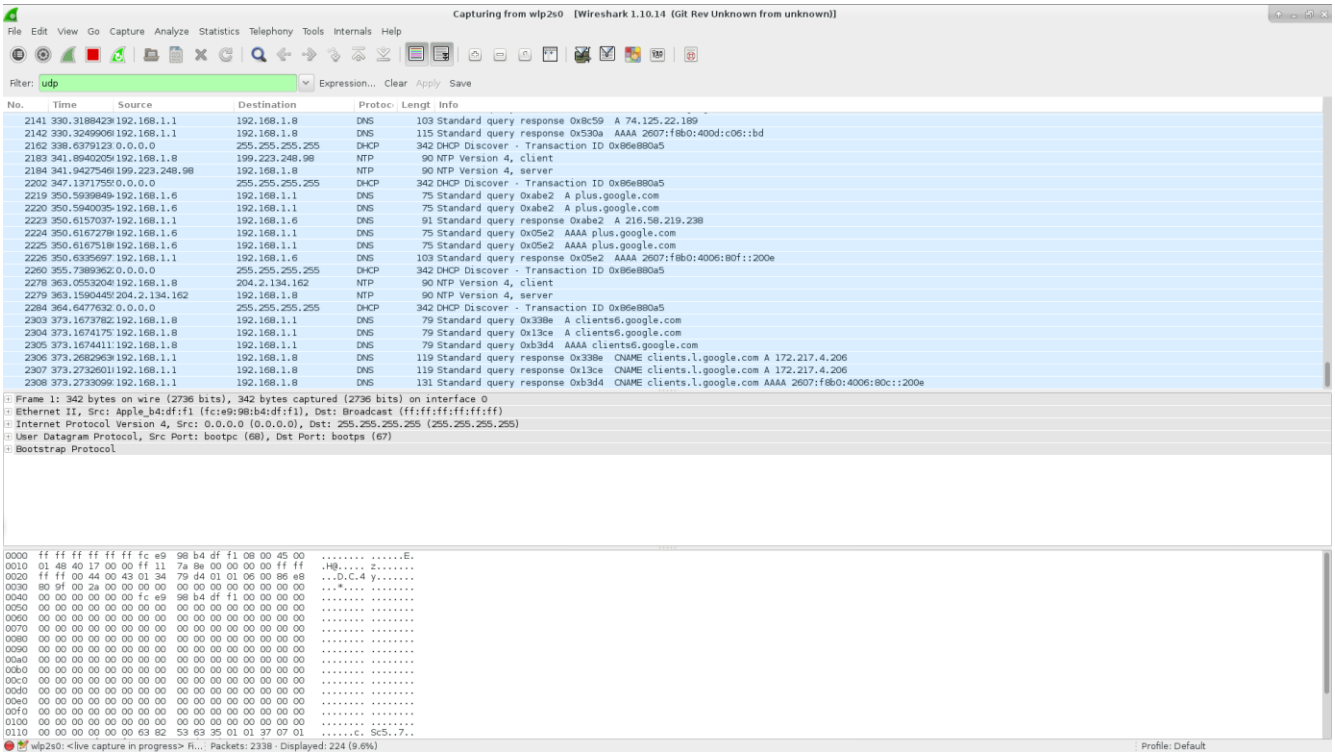
I show that I am only capturing IP packets.

Filtering by TCP:



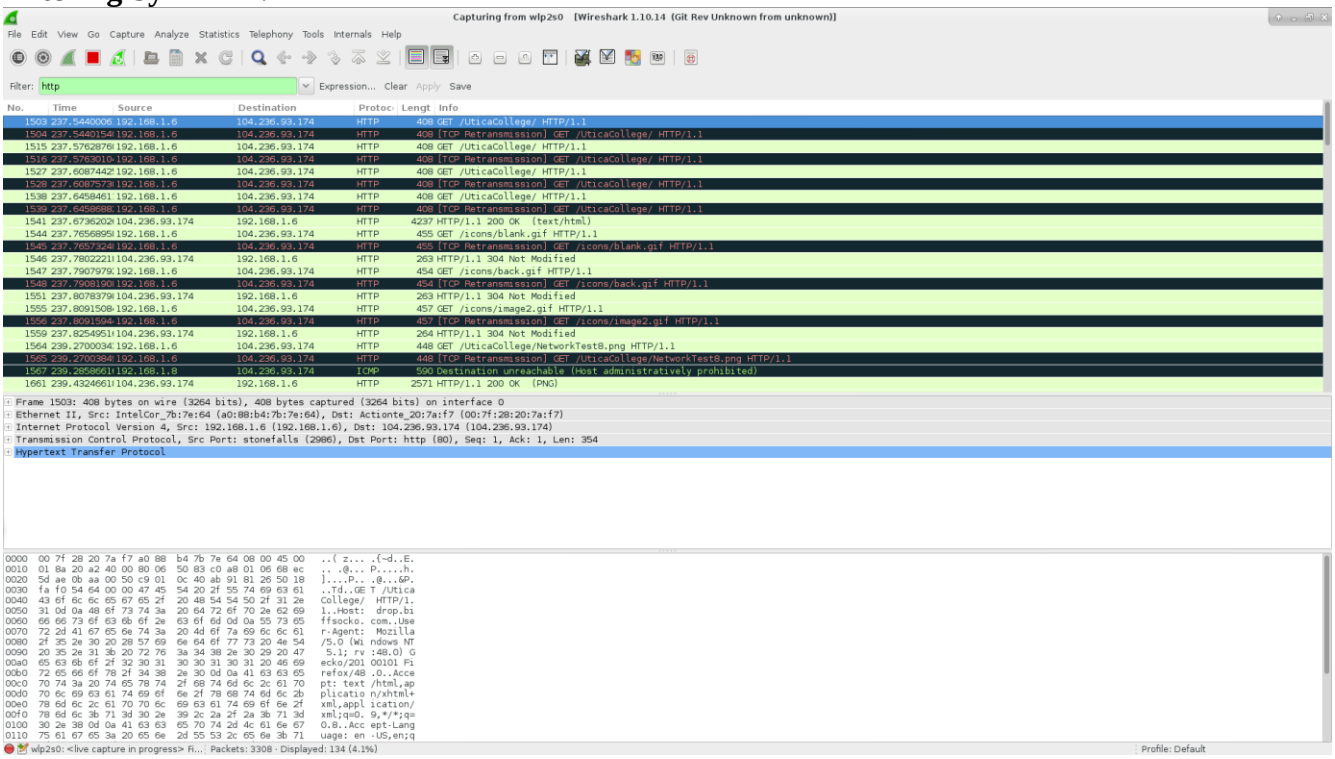
Filtering by TCP shows application layer data – perhaps the most interesting things I'm looking for from a cyber security perspective.

Filtering by UDP:



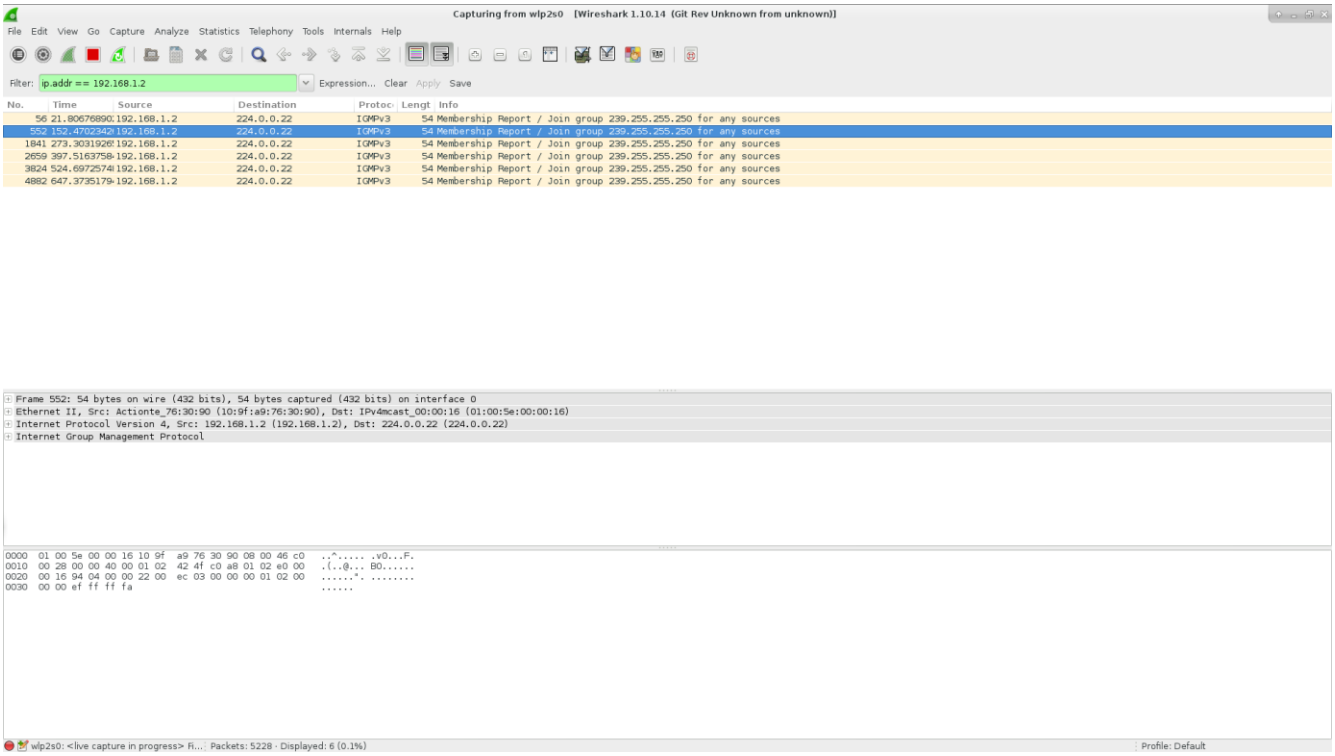
Filtering by UDP shows the expected UDP applications (NTP, DNS, DHCP, NFS etc).

Filtering by HTTP:



Filtering by “http” shows mostly http traffic. I did encounter an ICMP packet with this filter enabled, which was kind of weird. I have no explanation for this.

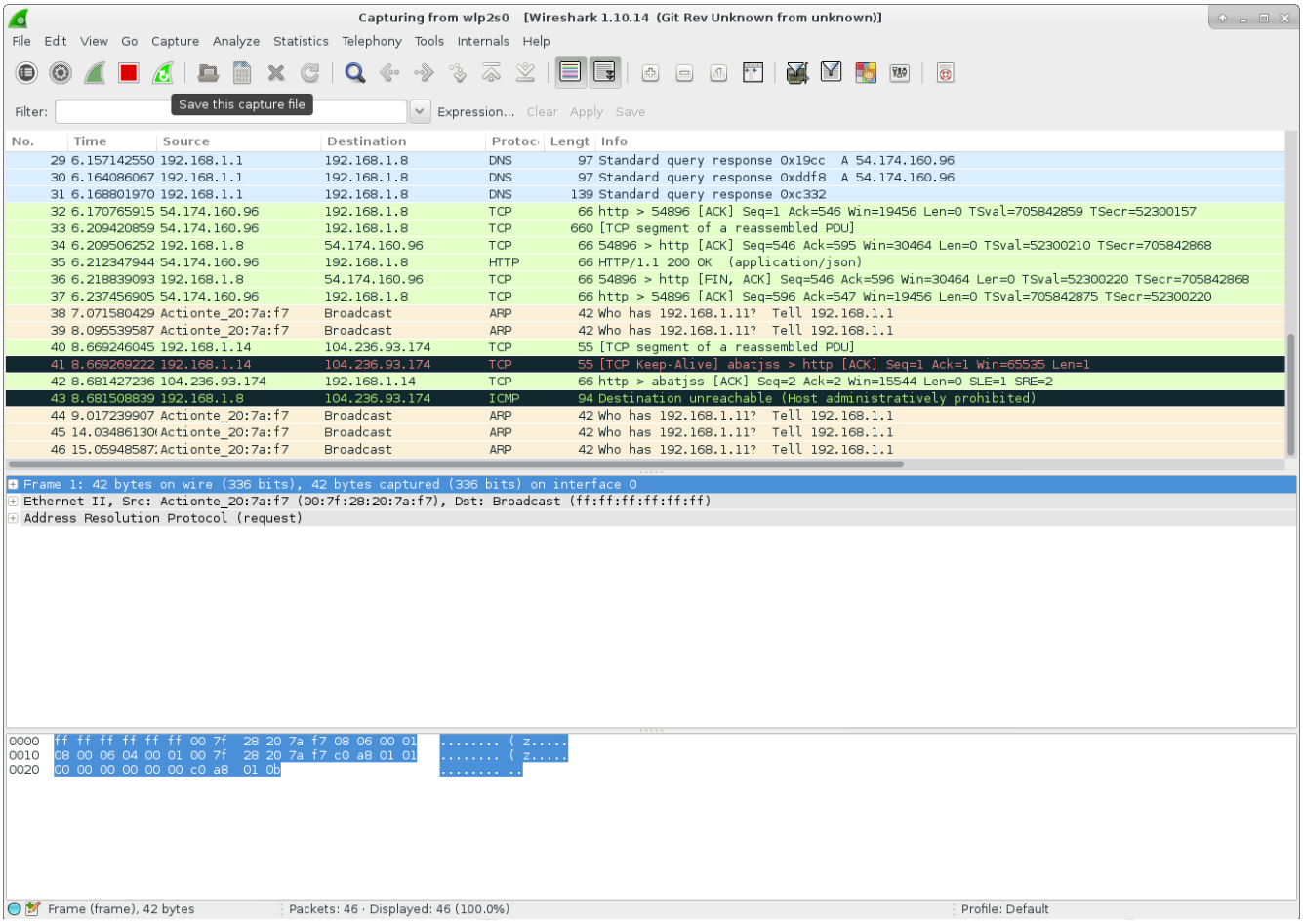
Specifying an IP address:



When filtering by IP address, I narrow down the packets that have as a source or destination the specified IP address.

View Packet Summaries with the Packet List Window

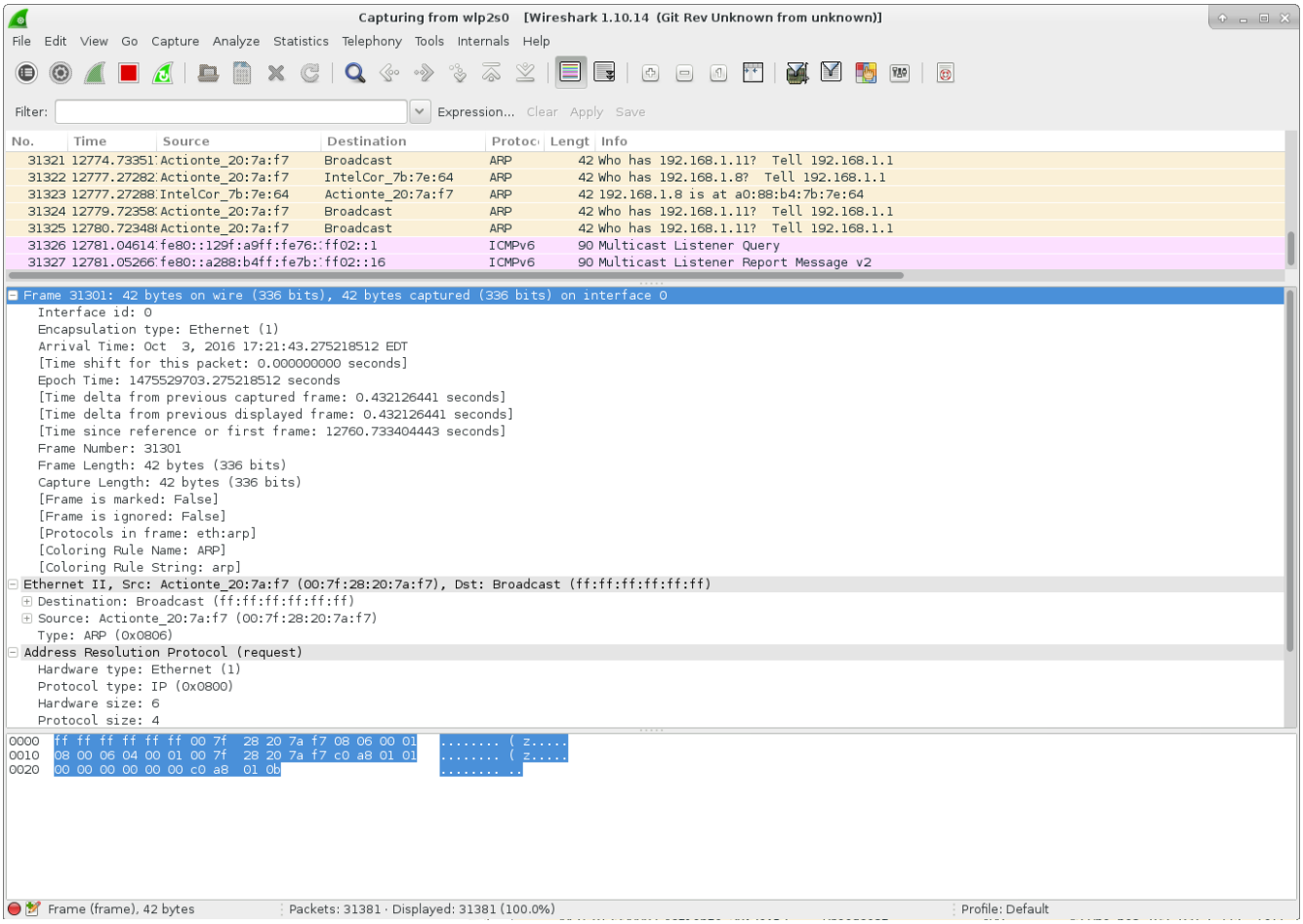
The packet frame in the below picture shows a number of columns.
Column 1 – shows the packet number
Column 2 - shows the time in a number of formats. In the example below, I have chosen the number of seconds since the live capture began. It is accurate to the nanosecond.
Column 3 - is the source IP address (where the packet is coming from)
Column 4 - is the destination IP address (where the packet is going to)
Column 5 - is the protocol
Column 6 - is the length of the packet
Column 7 - is information associated with the packet.



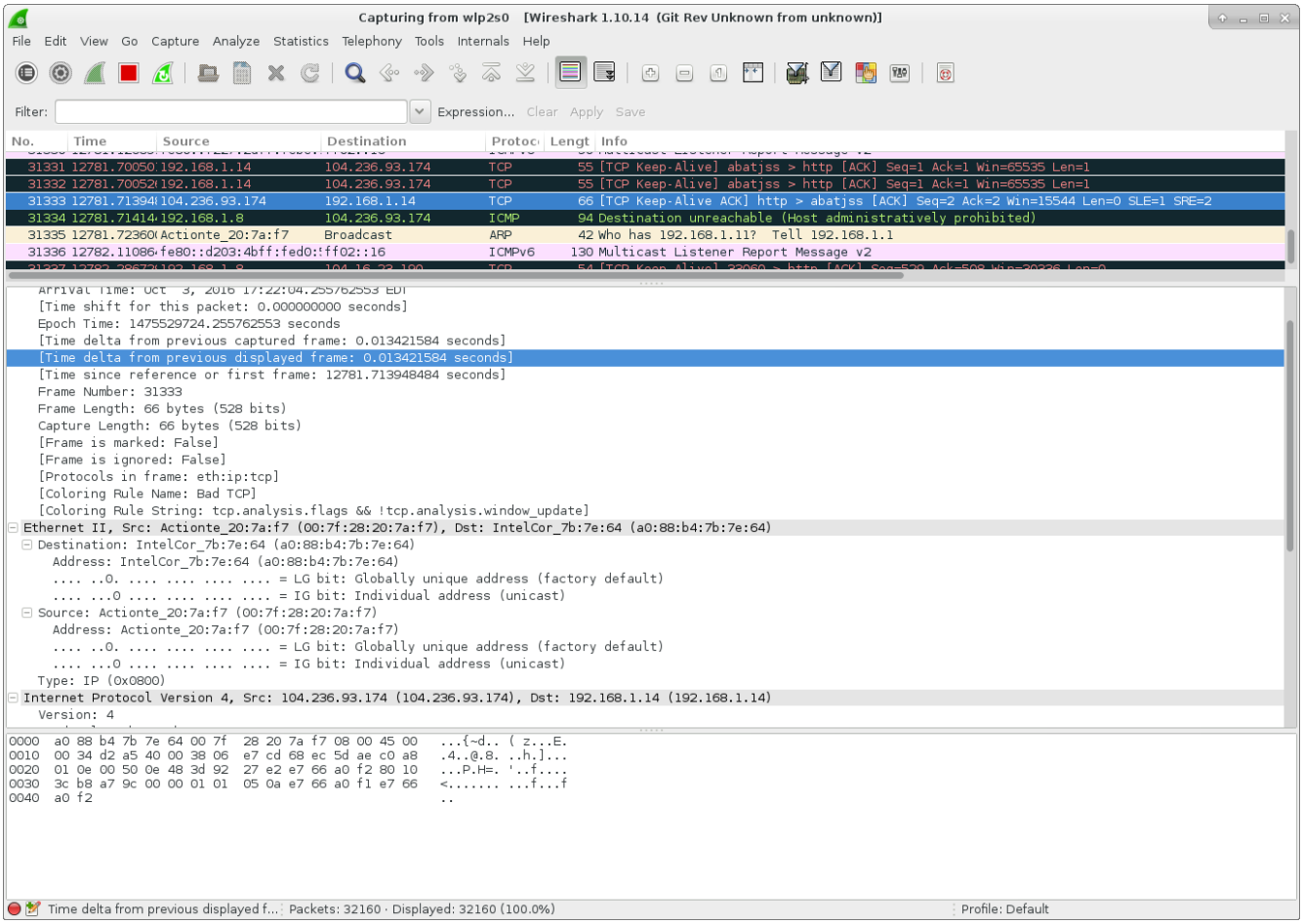
Study Packet Details with the Packet Details Window

By highlighting a packet in the packet summary window, you can view further information about that packet in the packet details frame.

Here are some of the details of an ARP packet:



And here are some of the details of a TCP packet:



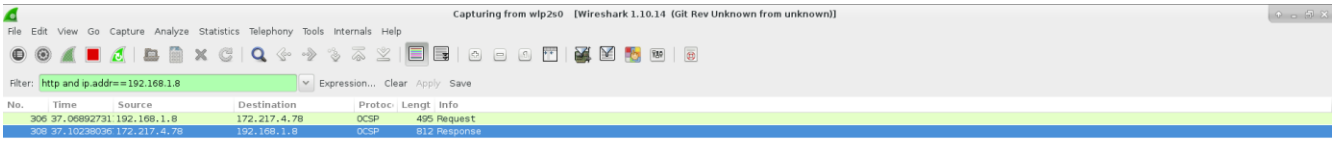
The information in the packets differs. Obviously the protocol is labeled differently, but also things like frame length, payload and the TCP packet gives information about what version it's using.

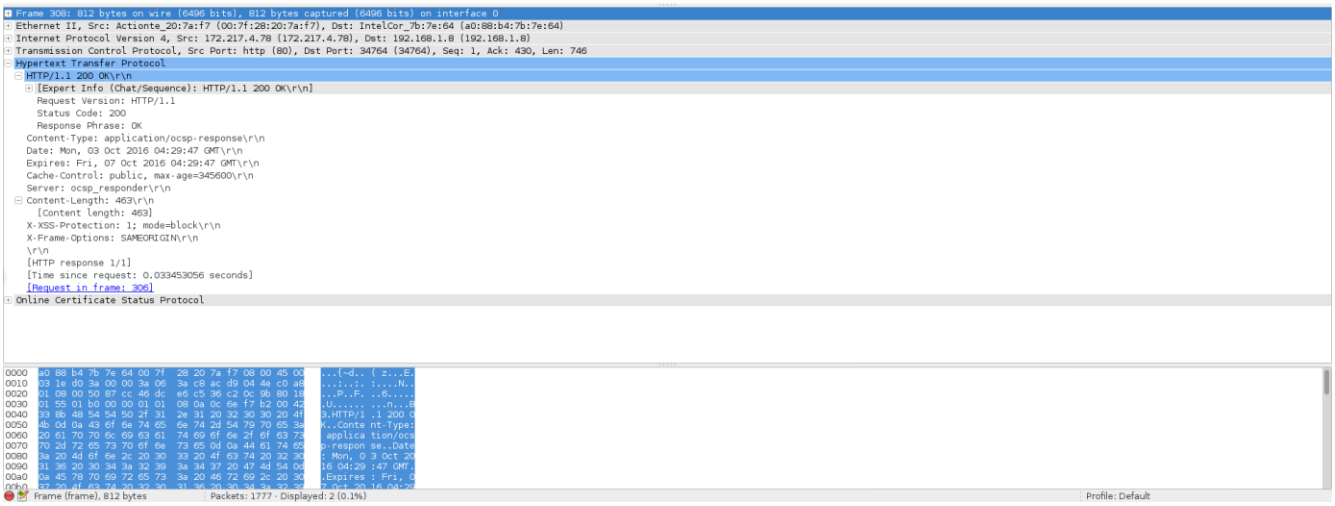
View Packet Data with the Individual Packet Bytes Window

Using the above examples, the Packet Bytes window is the bottom frame. The data held in the packet bytes window is different because both packets are different. The same as opening a file in a hex editor, the hexadecimal representation is on the left side of the frame, and its ASCII representation is on the right side of the frame. There is a period (".") for hexadecimal information that does not have ASCII characters associated with it.

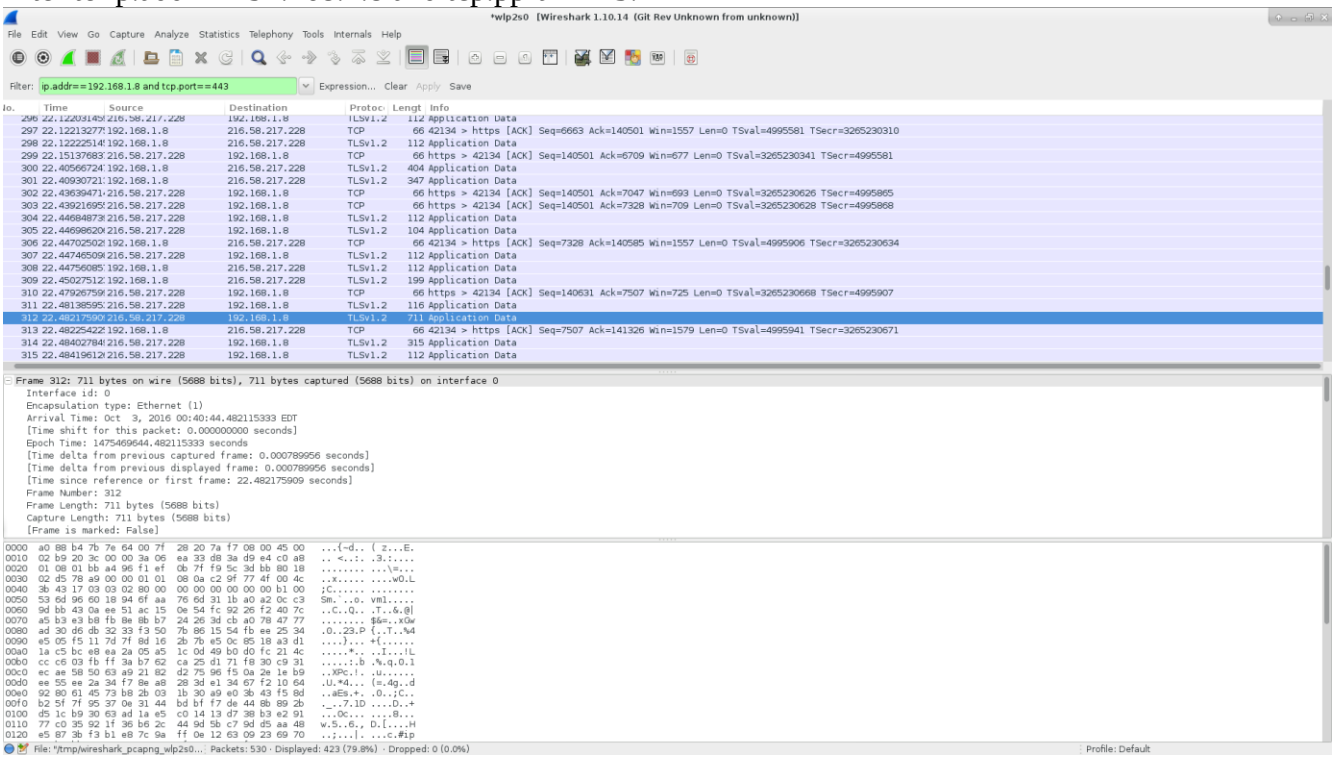
Browse The Internet

Filter on ip.addr==192.168.1.8 and http., visit htto://www.google.com and perform a search on scurvy

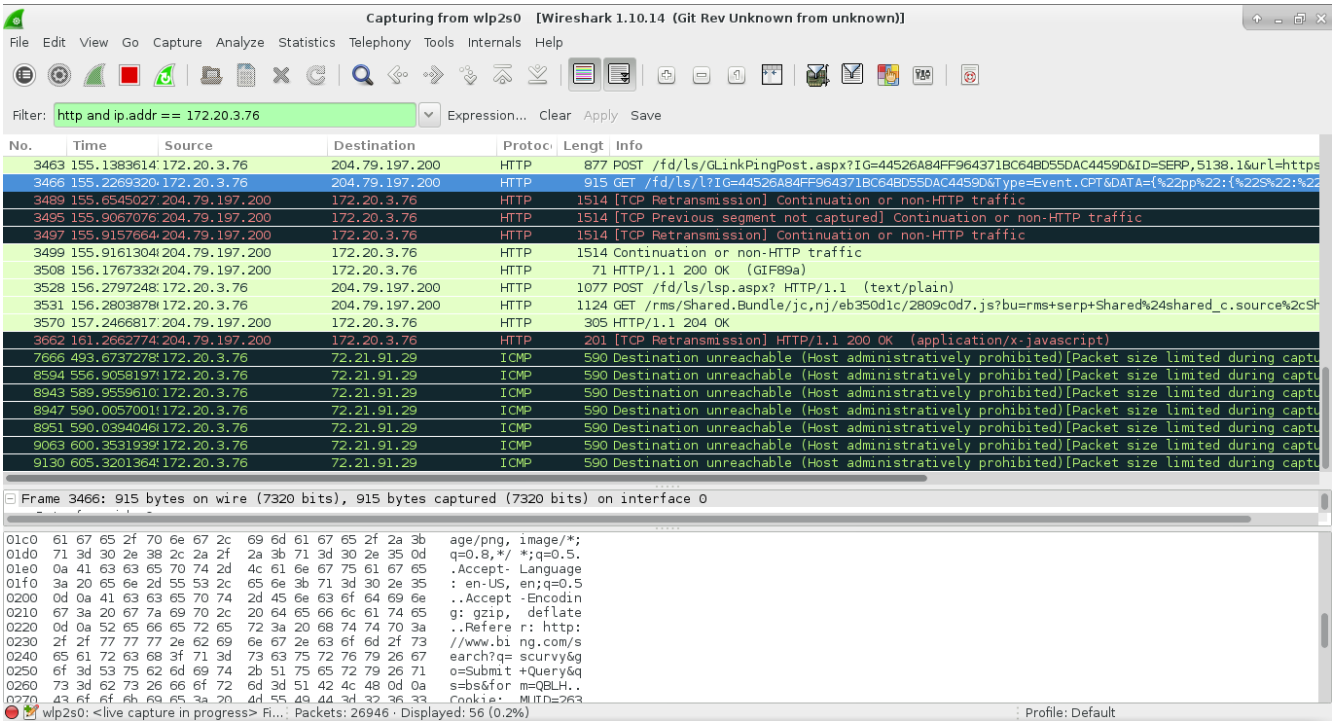




When this action was performed, I found that Google redirected to https – port 443. I then changed my filter to ip.addr==192.168.1.8 and tcp.pprt==443:



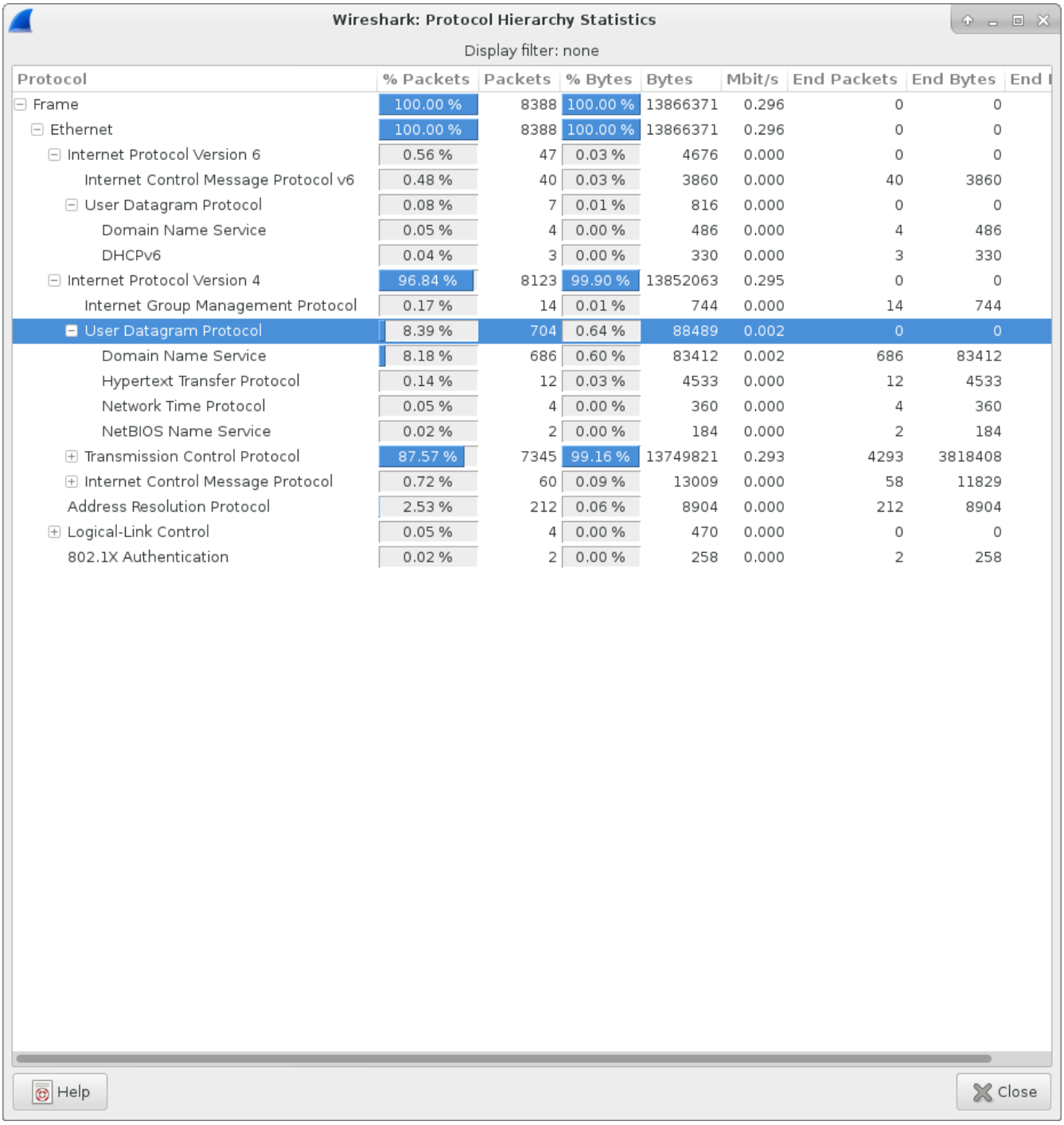
After inspecting the packets I found that my search on scurvy was in fact secure using the SSL connection. I then tried the same experiment with the Microsoft Bing search engine:



In this case I was able to view my search within the packet because it was going over a non-encrypted connection. I was also able to view data from the Wikipedia page (first link) as it too was non-encrypted.

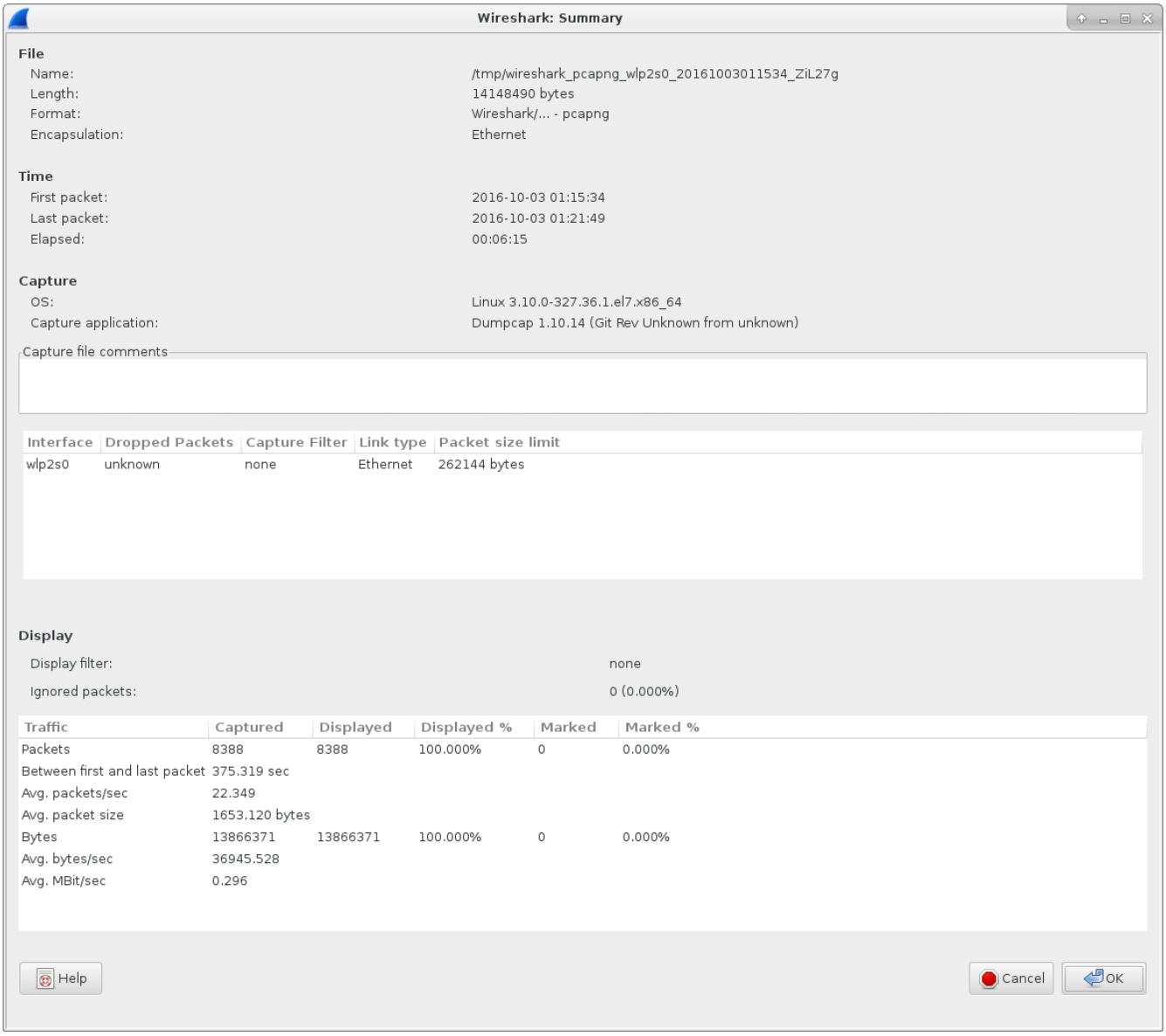
Analyze Wireshark Data

a) How many UDP packets did Wireshark capture: 704

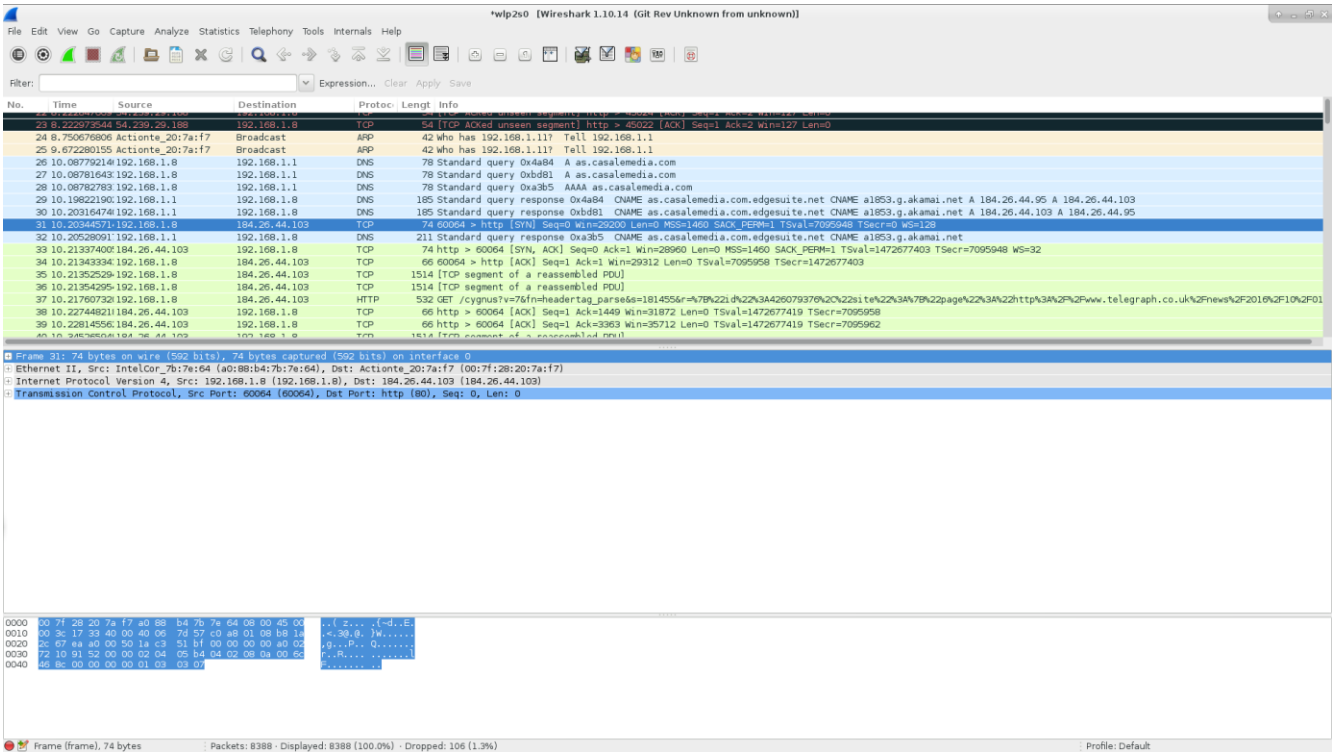


Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End I
Frame	100.00 %	8388	100.00 %	13866371	0.296	0	0	
Ethernet	100.00 %	8388	100.00 %	13866371	0.296	0	0	
Internet Protocol Version 6	0.56 %	47	0.03 %	4676	0.000	0	0	
Internet Control Message Protocol v6	0.48 %	40	0.03 %	3860	0.000	40	3860	
User Datagram Protocol	0.08 %	7	0.01 %	816	0.000	0	0	
Domain Name Service	0.05 %	4	0.00 %	486	0.000	4	486	
DHCPv6	0.04 %	3	0.00 %	330	0.000	3	330	
Internet Protocol Version 4	96.84 %	8123	99.90 %	13852063	0.295	0	0	
Internet Group Management Protocol	0.17 %	14	0.01 %	744	0.000	14	744	
User Datagram Protocol	8.39 %	704	0.64 %	88489	0.002	0	0	
Domain Name Service	8.18 %	686	0.60 %	83412	0.002	686	83412	
Hypertext Transfer Protocol	0.14 %	12	0.03 %	4533	0.000	12	4533	
Network Time Protocol	0.05 %	4	0.00 %	360	0.000	4	360	
NetBIOS Name Service	0.02 %	2	0.00 %	184	0.000	2	184	
Transmission Control Protocol	87.57 %	7345	99.16 %	13749821	0.293	4293	3818408	
Internet Control Message Protocol	0.72 %	60	0.09 %	13009	0.000	58	11829	
Address Resolution Protocol	2.53 %	212	0.06 %	8904	0.000	212	8904	
Logical-Link Control	0.05 %	4	0.00 %	470	0.000	0	0	
802.1X Authentication	0.02 %	2	0.00 %	258	0.000	2	258	

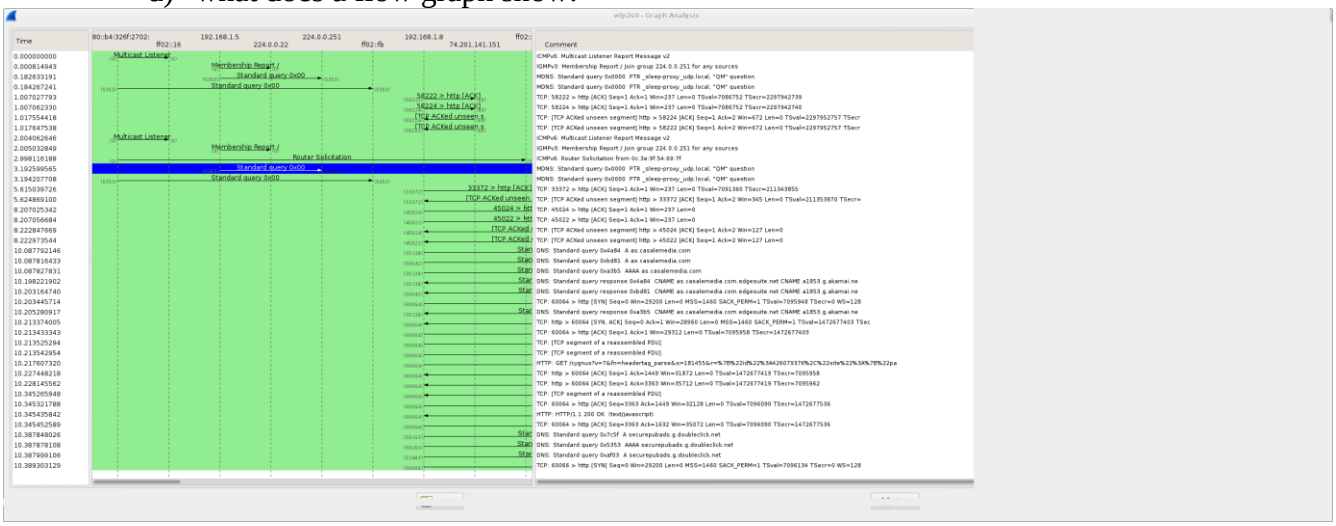
b) what was the average IP Packet size: 1653.120 bytes



c) how many packets did Wireshark drop:106

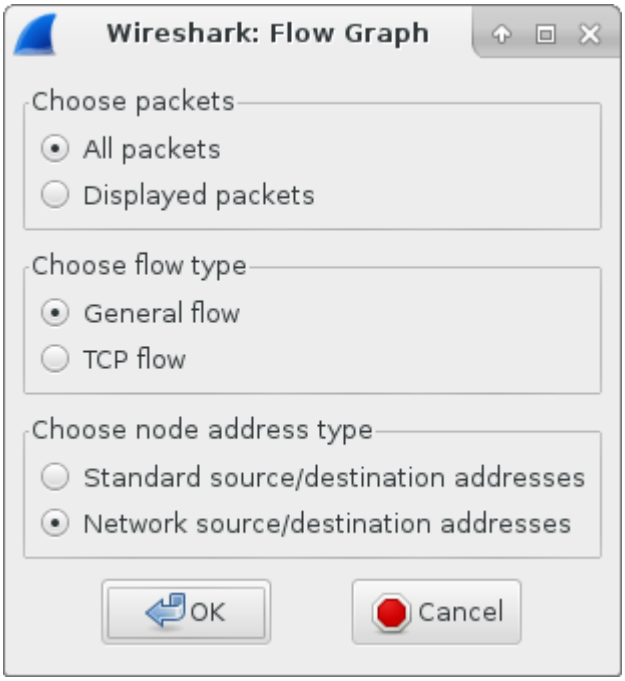


d) what does a flow graph show:



A flow graph shows the data flow of a connection. By scrolling to the right, I can see things like retransmits or drops.

e) list the flow graph options



Analysis Questions

1. How can security professionals use Wireshark?
By using Wireshark, you can view the raw data of a packet for file signatures we discussed in our hex lab, or spot instructions for execution of malicious code. Another way to use Wireshark is to discover DoS attacks or ssh brute force attacks. By capturing an overwhelming number of a certain type of incoming packets you can take steps to block the source IP address. If I saw a large number or packets going to a remote destination IP address, I would investigate to make sure corporate data was not being removed from an unauthorized source.
In addition, I could monitor incoming and outgoing traffic based on protocol and create rules. For SMTP as an example, I could monitor for attachments of a certain size or content.

2. List three ways attackers can use Wireshark.
1) To map the network
2) Discover protocols that are in use
3) Find clear text communications for username and password discovery

3. Does Wireshark capture all the traffic on the Internet? If so, explain why. If not, which traffic does it capture?
No, Wireshark captures traffic that passes its network interface either directly or on a hub, as well as broadcast and multipath traffic (Weadock, 2009).

4. Write Wireshark filters to:
a) View all traffic for 10.10.10.2. Filter = ip.addr==10.10.10.2
b) View icmp traffic from any address. Filter = icmp

5. Has this lab changed your perspective on your privacy while browsing the Internet? Describe why or why not.

It hasn't changed my perspective of privacy on the Internet; because of my job, I've been exposed to packet sniffers and understand their capabilities. That said, this lab gave me an opportunity to dive in a lot deeper than I had previously and look at some of the features that Wireshark has. So, basically I have a more, well rounded understanding of packet sniffing and filtering then I had previously.

Conclusion

Through the exercises, I learned about the capabilities of the Wireshark packet sniffer. I was able to capture packet data and inspect individual packets. I gained a better understanding about the contents of a packet, and how I can gather useful information from using the Wireshark sniffer and putting what I learned into the context of how a network is used and what kinds of data I'm expecting to see.

References

Adrian Hannah (2011 Nov. 14) Packet Sniffing Basics. Retrieved from http://www.linuxjournal.com/content/packet-sniffing-basics

Glenn Weadock (2009 Sept 30). Wireshark and Promiscuous Mode - Why you may not be seeing all the traffic you think you should. Retrieved from http://www.networkworld.com/article/2231903/microsoft-subnet/wireshark-and-promiscuous-mode.html

Paessler (1996) "Packet Sniffing; Packet Sniffing With PRTG Network Monitor" Retrieved from https://www.paessler.com/packet_sniffing

Craig Hunt (1992). TCP/IP Network Administration. Sebastopol, CA: O'Reilly & Associates

Douglas E. Comer (1997). Computer Networks and Internets with Internet Applications; Third Edition. Upper Saddle River, New Jersey. Prentice Hall

OpenBSD Documentation (2016 Sept 1), PF Users Guide, Retrieved from https://www.openbsd.org/faq/pf/filter.html

W. Richard Stevens (1990). UNIX Network Programming. Englewood Cliffs, NJ. Prentice Hall

Sahin Erbay 2016, July 26) Internet Protocal Suite (TCP/IP and OSI Models); Retrieved from http://sahinerbay.com/2016/02/26/internet-protocol-suite-tcpip-and-osi-models/

Wireshark (2008 April 12) Duplicate Packets. Retrieved from https://wiki.wireshark.org/DuplicatePackets

Wireshark (2014). Wireshark User's Guide. Retrieved from https://www.wireshark.org/docs/wsug_html_chunked/