Wireshark - Packet Capture

Using Wireshark to capture packets on a certain interface and using filters to observe web traffic. Identifying problems in applications and networks by examining TCP/IP packets to and from servers

Introduction

Hello everyone! My initial interest in Wireshark was driven by concerns about potential cybersecurity threats to my home network. Specifically, I encountered an issue where my computer exhibited erratic behavior, such as unexpectedly powering on and off, and logging in without user intervention. Additionally, I discovered an unfamiliar user account within my computer's directory and noted that the Wake-on-LAN feature was enabled, whereas it is typically set to the default off position. I have decided to take on a guided project to broaden my understanding of network traffic to instill a sense of security and prevent future threats.

Scope and Objectives

The project basis will include multiple tasks which will include guided project pictures and Linux commands.

Tasks/Objectives:

- Install and setup Wireshark on Ubuntu - (Alternative - Use Kali Linux)
- Start a packet capture on an ethernet port and save it to a file
- Use display filter to detect HTTPS packets
- Visit web page and detect its IP address using a display filter
- Locate all HTTPS packets from a capture not containing a certain IP address

Task 1 - Install and setup Wireshark on Ubuntu

To get the latest stable version of Wireshark on Ubuntu Linux, we have use the add-apt-repository command

Which would look like:

```
sudo add-appt-repository ppa:wireshark-dev/stable
```

[***Note:*** Wireshark should not be run as superuser (sudo) for security reasons]

The user can be added to the Wireshark group to add packet capture capabilities
(This uses a set of Linux commands: `sudo, usermod, -aG`)
`Sudo` - Superuser Do
`Usermod` - User modification
`-aG` - Add to Group/Append Group

This would look like this:

```
sudo usermod -aG wireshark $USER
```

Task 2 - Start a packet capture on an ethernet port and save it to file:

The wired interface includes the ethernet packet capture, which begins with ‘en’ in Wireshark
The Wireshark app includes controls to start packet capture, stop capture, save the packets to a file and load the capture file
[***Note***: A capture can only be saved once the capture has stopped]

Task 3 - Use a display filter to detect HTTPS packets

To display certain packets in an existing packet capture, use a display filter
To display only HTTPS traffic, use a filter on `TCP port 443: tcp.port == 443`

Task 4 - Visit a web page and detect its IP address using a display filter

A TLS handshake display filter may be used to detect a website visit in a packet list:
`tls.handshaketype == 1`
The IP address is used in a filter to obtain packet information for a particular website:
`ip.addr == 142.251/163/105`

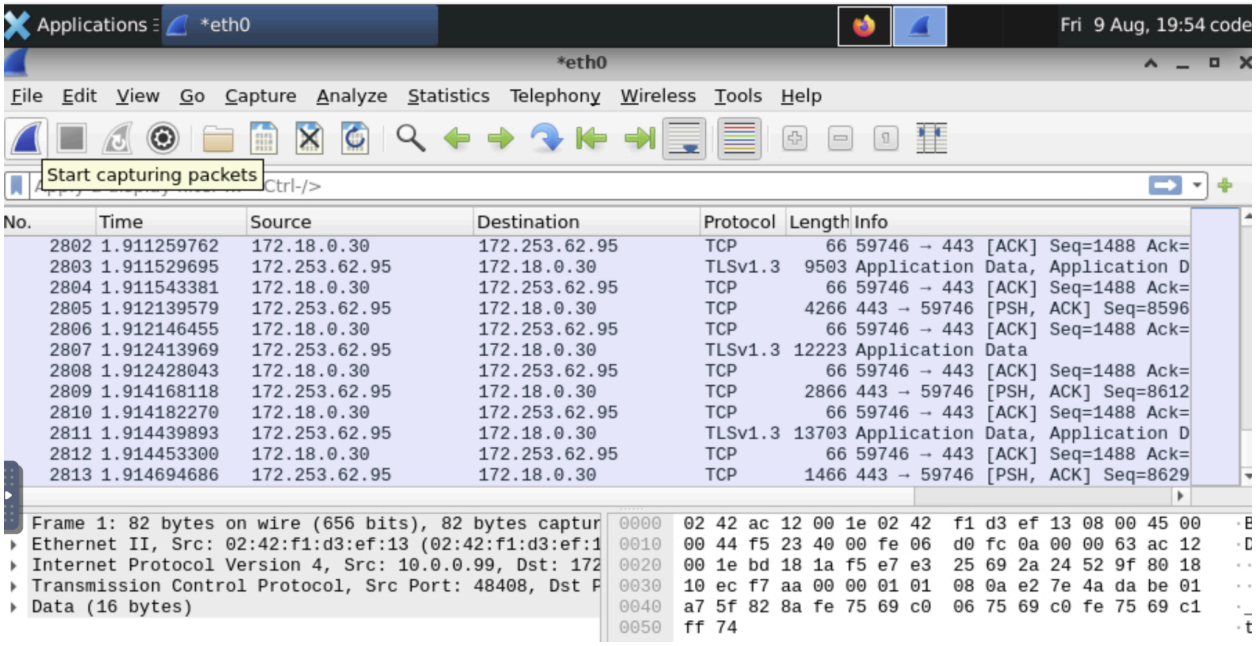
Task 5 - Locate all HTTPS packets from a capture not containing a certain IP address:

A conditional statement may be used to include and eliminate packets from a Wireshark capture:
`!(ip.addr == 8.43.85.97)` and `tcp.port == 443`
A compound conditional should include parentheses to avoid order of execution errors:
`!(ip.addr == 8.43.85.97)` and `(tcp.port == 80 or tcp.port == 443)`

Further Solidification Task

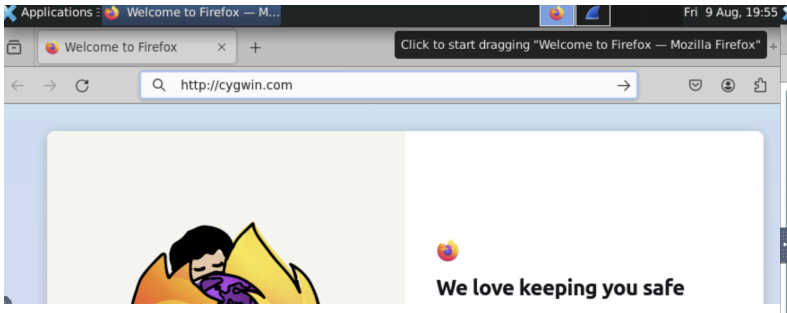
To further solidify my knowledge on the topics discussed in this project - I have been given a separate task
I will be capturing ethernet traffic using Wireshark while visiting a site that uses HTTP (TCP port 80)
I will be applying a display filter to view only HTTP traffic and save the capture to a file

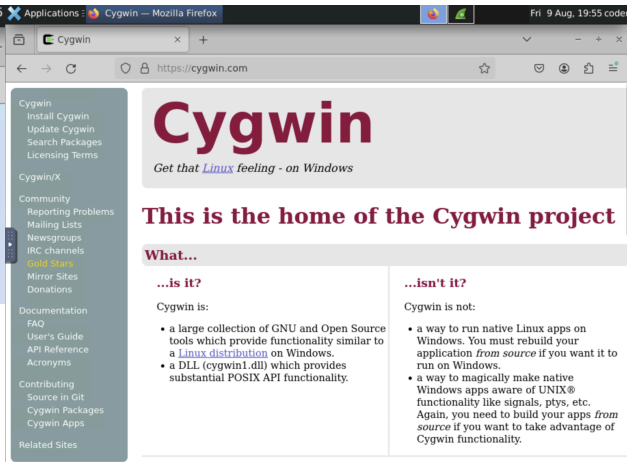
Task 1 - In Wireshark, start capture



This task initializes the capture on Wireshark, In the image above there is a blue shark fin button which starts capture.

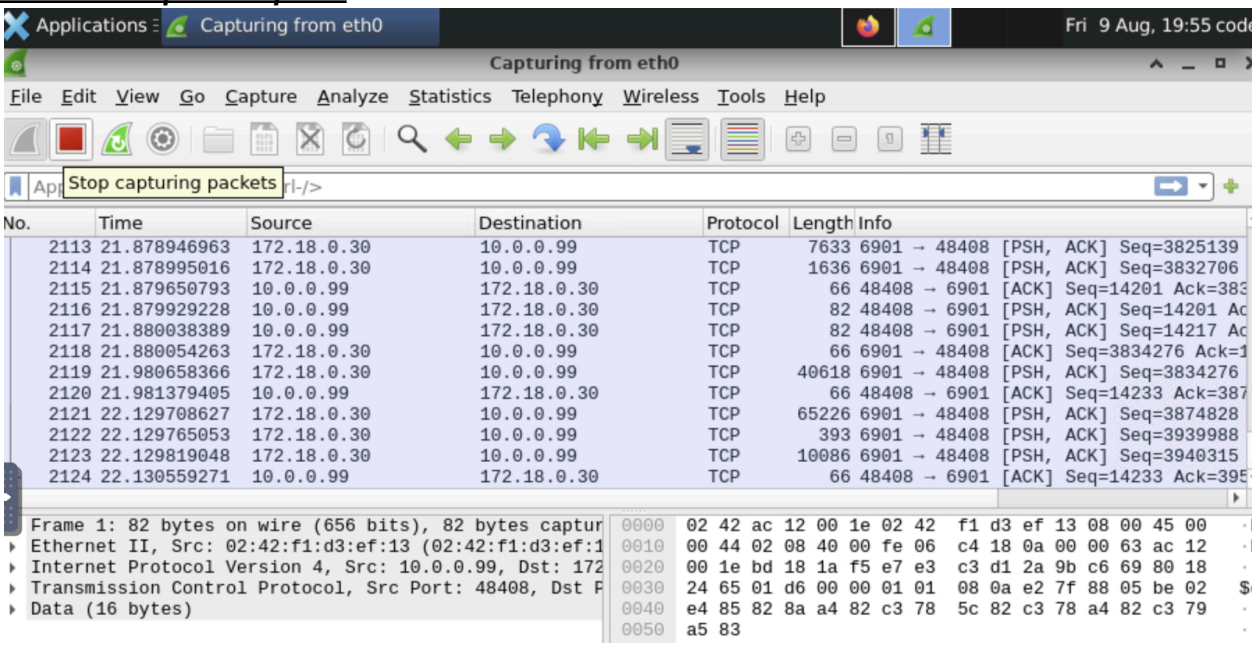
Task 2 - Visit the URL http://cygwin.com in the Firefox browser window





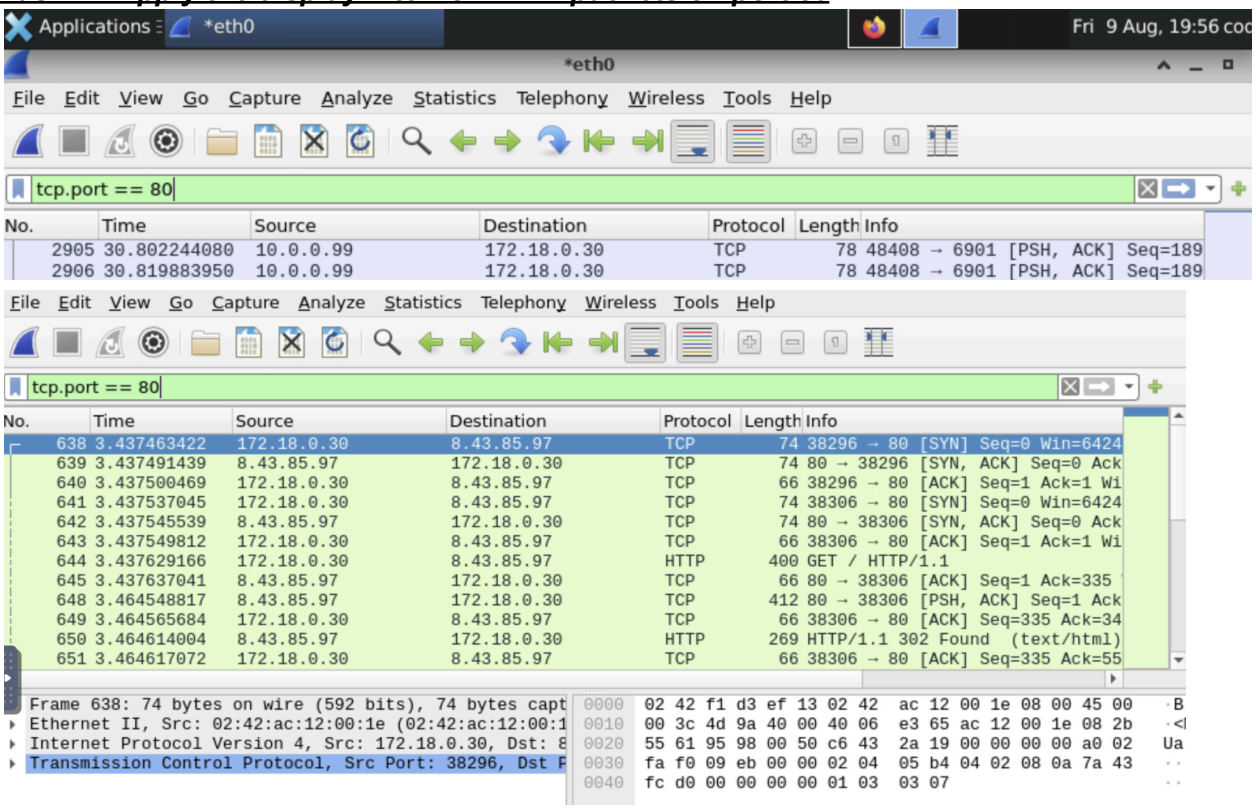
While the Wireshark program is currently running, you would want to head on over to the url using FireFox browser. This will capture all packets communicated within the network on our Wireshark application.

Task 3 - Stop the capture



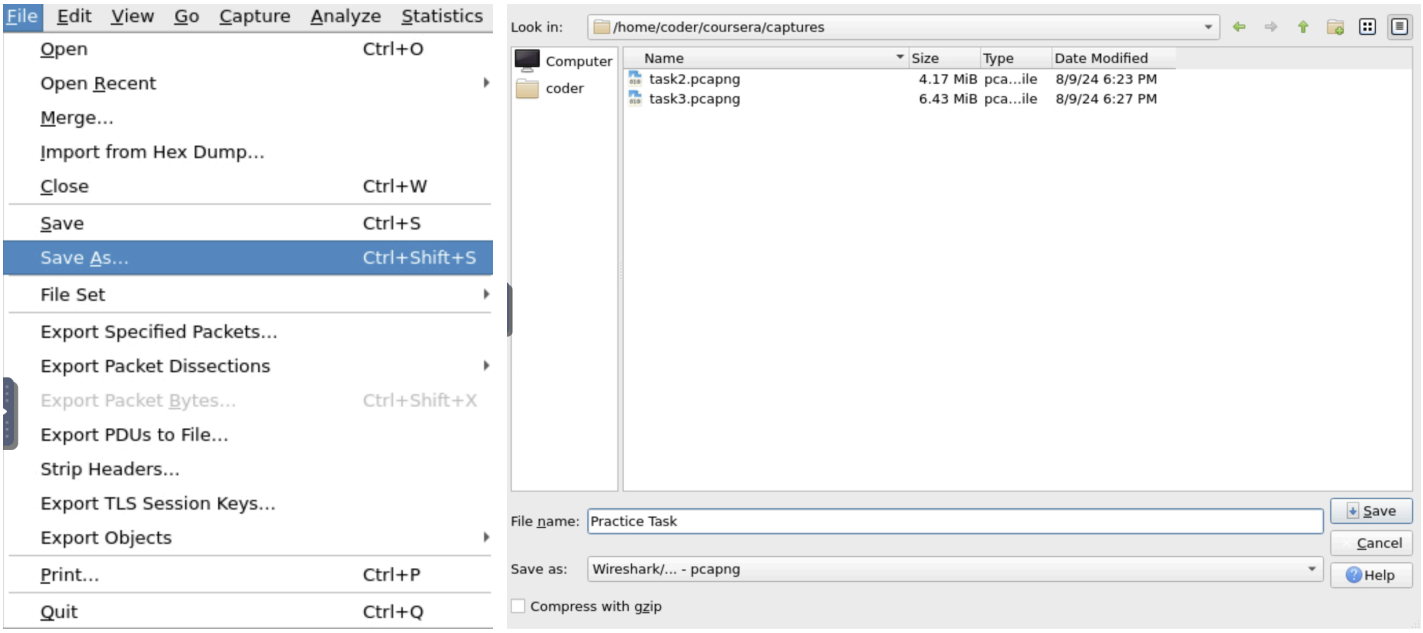
The image above shows a red button right next to the blue shark fin button we pressed earlier. This red button stops all capture from the network.

Task 4 - Apply the display filter for HTTP packets on port 80



When searching for HTTP packets only we would need to apply a display filter in the Wireshark application. This would be found in the search box area right below the toolbar. In here you would want to type in `tcp.port == 80` (This is because tcp port 80 is used for HTTP traffic)

Task 5 - Save the file

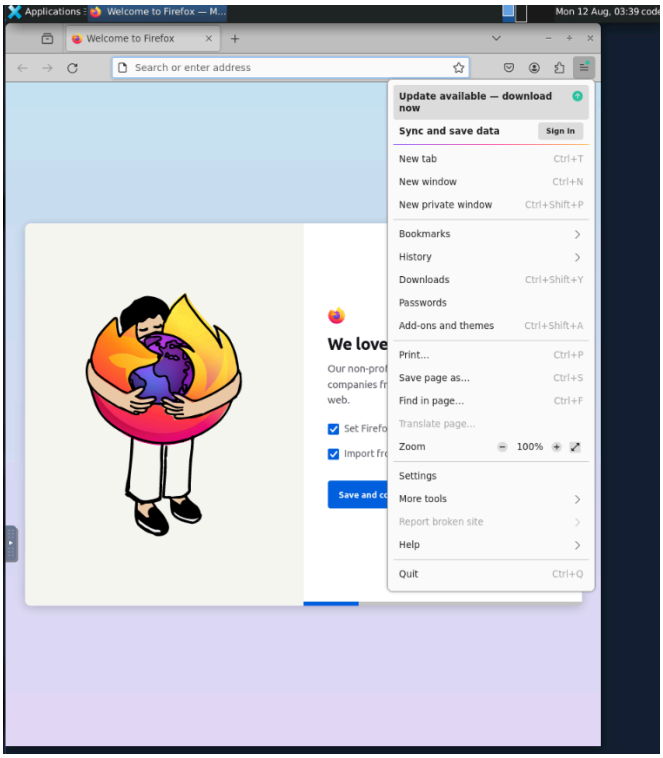


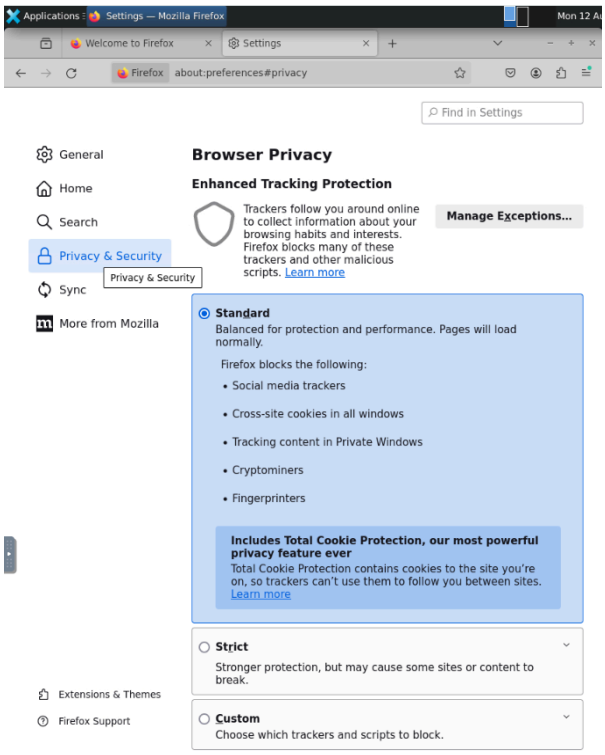
Saving the file can be done in two ways. However, for this demonstration, going into the File button on the top left hand corner, we are able to find the “save as” option, which we will be using to create a file called “Practice Task”. Note that in Wireshark, the files will be saved as a “.pcapng”

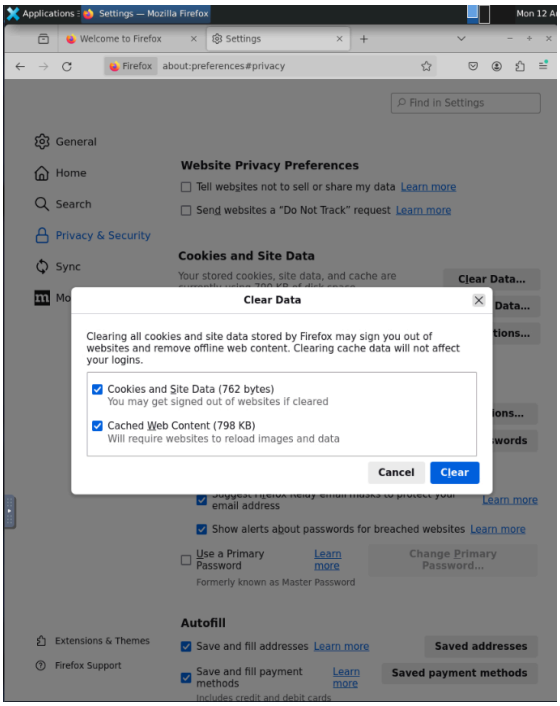
Capstone Task

Capstone Task - Use Wireshark to capture and observer ethernet packets on HTTP and HTTPS ports

Task 1 - Clear the Cache in the firefox browser

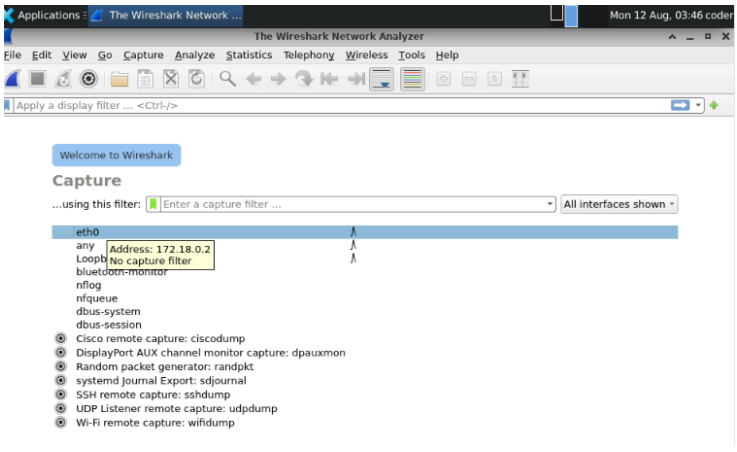


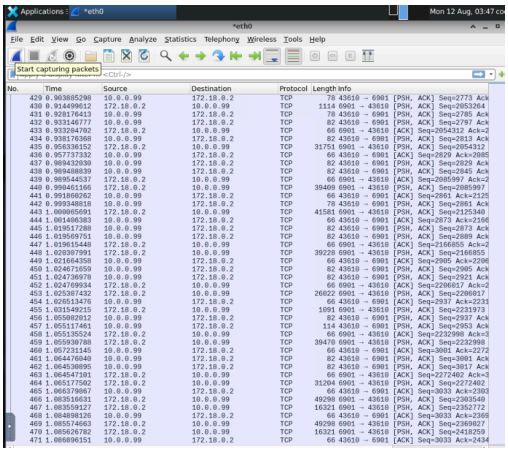




In this first task, we are to clear the cache of our browser before we capture web traffic.
To do so we must first open the hamburger icon on the top right hand of the screen. Once in the location, we are to open the settings tab using the settings option. We then head on over to the Privacy and Security tab where we will be able to find the Cookies and Site Data menu. We can click on the "Clear Data" button to clear all cookies and cache for mozilla fire fox.
The 3 pictures presented are placed to help complete this task.

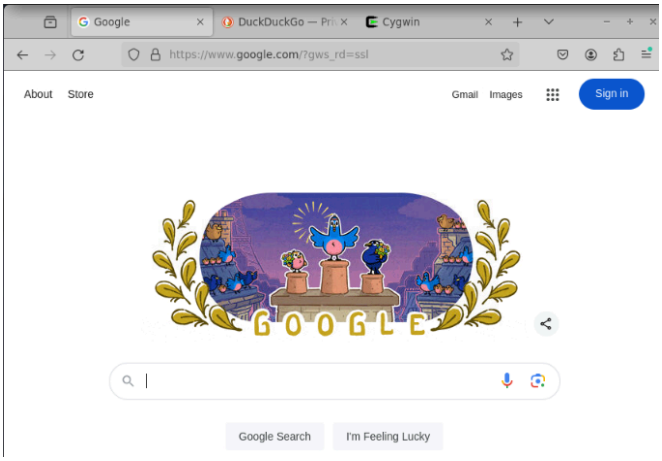
Task 2 - Start a packet capture on the ethernet in Wireshark

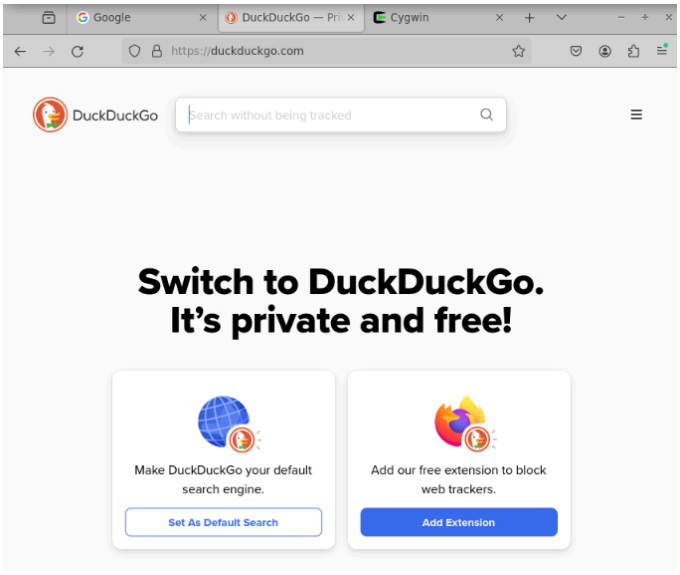


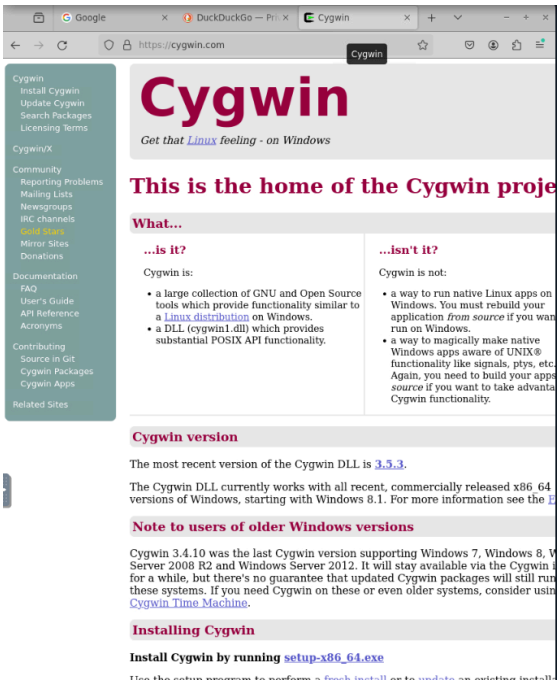


When opening Wireshark, you are presented with multiple network options to choose from. We are to use the eth0 which stands for Ethernet to complete the following tasks.
To begin scanning we can click on the Blue shark fin on the top left hand corner.

Task 3 - Visit Google.com, Duckduckgo.com, and http://cygwin.com (NOT HTTPS)

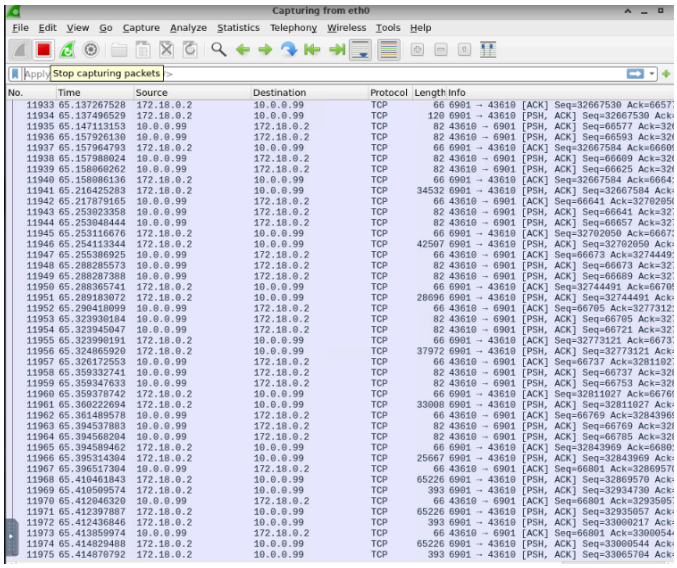




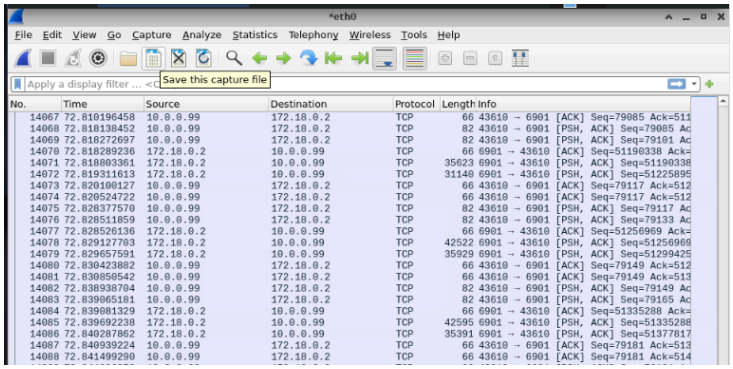


Visiting each site is not very difficult. Each website has a URL which can be placed in the search bar in the browser. For this exercise we are to use the Mozilla Firefox browser where I was able to type in the URLs: google.com, duckduckgo.com, and http://cygwin.com.

Task 4 - Stop the packet capture in Wireshark and save the capture to a file

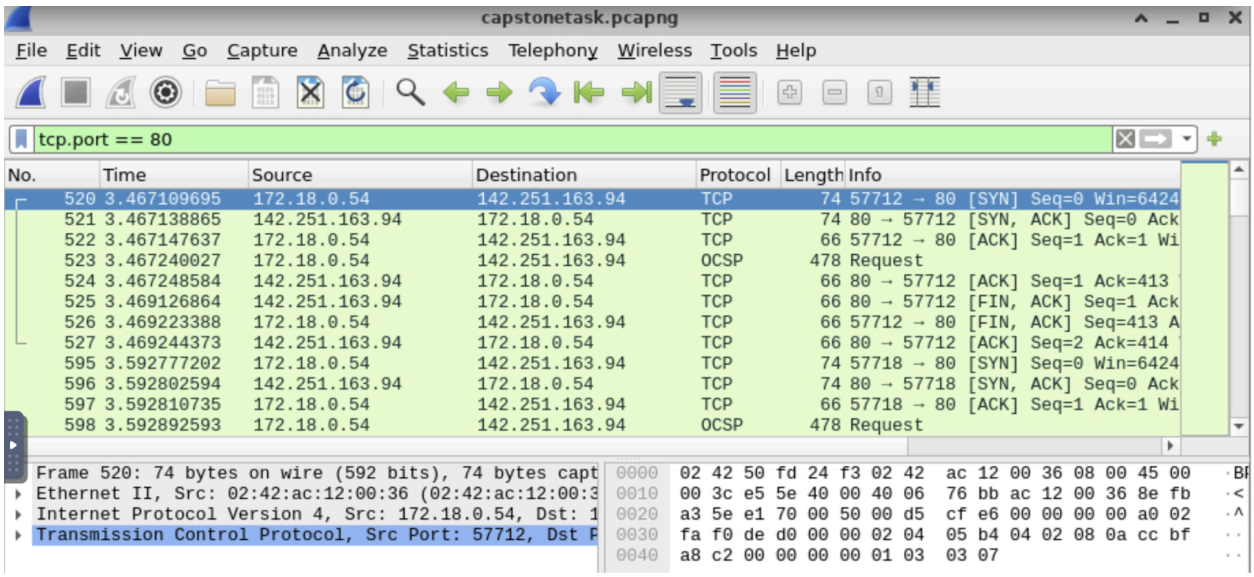


To stop the capture from Wireshark, We can press the Red Square button, next to the Blue Shark Fin.



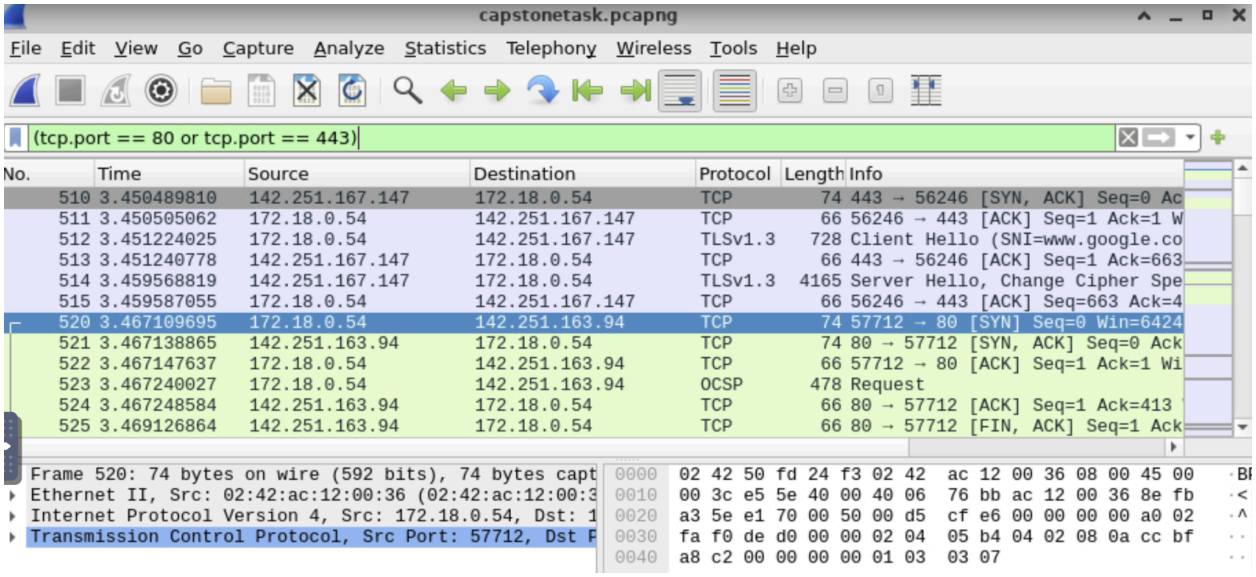
However, to save the file we can click on the “File” button or the first clipboard button that is placed beside the Folder icon. This will enable us to save our capture file in a desired location

Task 5 - Create a filter to just display port 80 TCP data. (IP grab of cygwin)



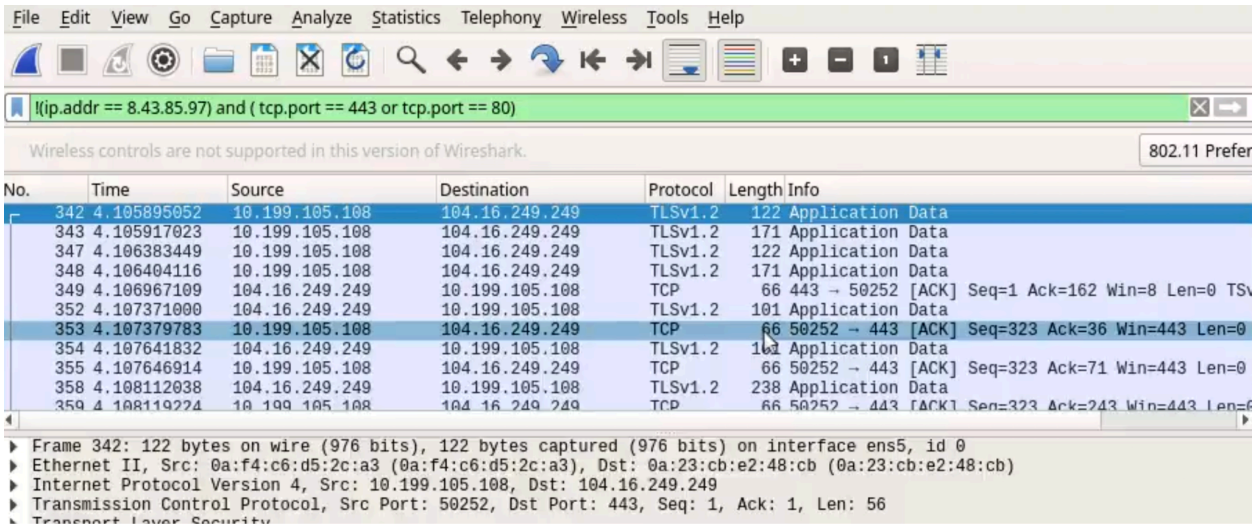
In the image above, we are able to see that there is a display filter added to the query. The query `tcp.port == 80` only shows the network traffic of HTTP. This allows us to search for a GET HTTP capture which would contain the ip address of the url http://cygwin.com.

Task 6 - Next, Create a filter to display only HTTP and HTTPS packets (Use OR condition)



In the image above, I was able to place down both HTTP and HTTPS packets by using the parentheses [`()`] and the "`or`" condition to display both ports. The query: `(tcp.port == 80 or tcp.port == 443)` allows us to see both HTTP and HTTPS packets.

Task 7 - Eliminate the Cygwin site visits from the displayed packets (Use AND condition)



In our final exercise, I would need to eliminate the cygwin ip address from the displayed packets. In order to do so, I would have to use the "`and`" condition as well as the [`!`] symbol. The [`!`] symbol is considered a "not" which would eliminate a certain parameter after the symbol has been placed. To use both effectively, we would have to place the query:
`!(ip.addr == 8.43.85.97) and (tcp.port == 443 and tcp.port == 80)`
The reason we are using the `ip.addr == 8.43.85.97` is because that IP address is the cygwin http ip address which was found in our initial scan from `tcp.port == 80`. So effectively the [`!`] is telling Wireshark "Do not include IP address `8.43.85.97`"

Lessons Learned

During this project, I have learned many new skills involving Wireshark. I have found uses in new conditions such as: (`ip.addr, ip.dst, ip.src`)(`or, and`)(`tls.handshake.type == 1`)(`tcp.port == 80 and 443`)

I was able to broaden my knowledge on HTTP and HTTPS as well as the use cases for all networks within my device. It also helped me understand how ip addresses are grabbed and how it is used between device to server communication.