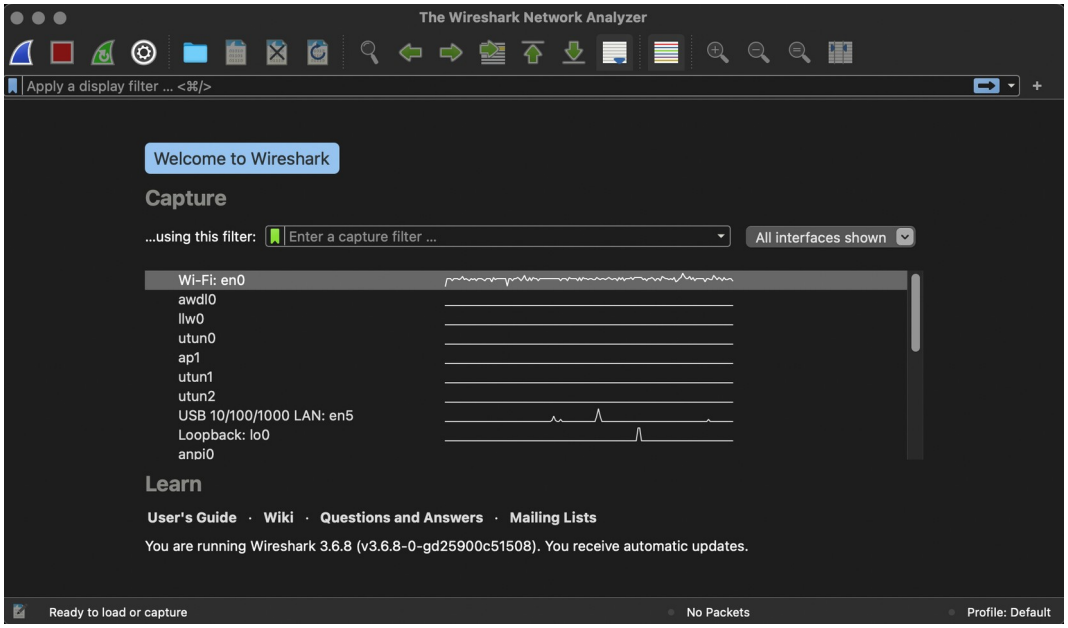
Name: Yu, Marco Jalen D. Section: S17

## 1.0 Laboratory Activity 1 – Investigating Encapsulation

Hello Class, for this laboratory activity we would be exploring the concepts of data encapsulation and decapsulation by investigating the contents of network packets. In order to do this, let us first download Wireshark, which is a well-known and widely used network protocol analyzer, allowing us to inspect and investigate the data traffic that is happening on our network through the following link: https://www.wireshark.org/download.html

After downloading the appropriate stable release version for your operating system, don't forget to install the application on your machine as well as its dependencies by following the instructions provided during installation. Once you have Wireshark installed correctly, startup the application and verify if Wireshark could see the active network interfaces of your machine similar to the image provided below:

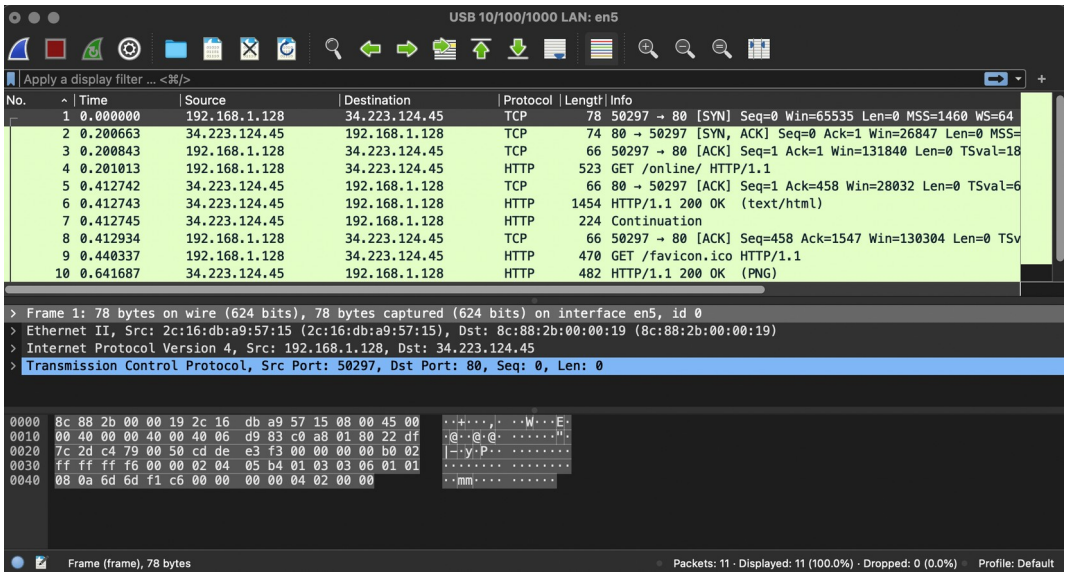



The sample image above shows initial page when you open Wireshark. In this page, you would need to select which network interface Wireshark would be capturing packets. The example provided enumerates multiple interfaces, however the important ones in the sample provided are the **Wi-Fi: en0, LAN: en5**, and the **Loopback: lo0**. Take note that your machine would most likely have different interfaces, interface names, and interface numbers from the snippet provided so don't worry too much about it. The **Wi-Fi** interface is just basically the wireless interface of your computer if you have it, the **LAN** interface is your typical network card where you plug-in your LAN port, and the **Loopback** is just a virtual interface of your computer allowing your computer to send messages to itself, which feels weird but it actually fills up an important function of your computer.

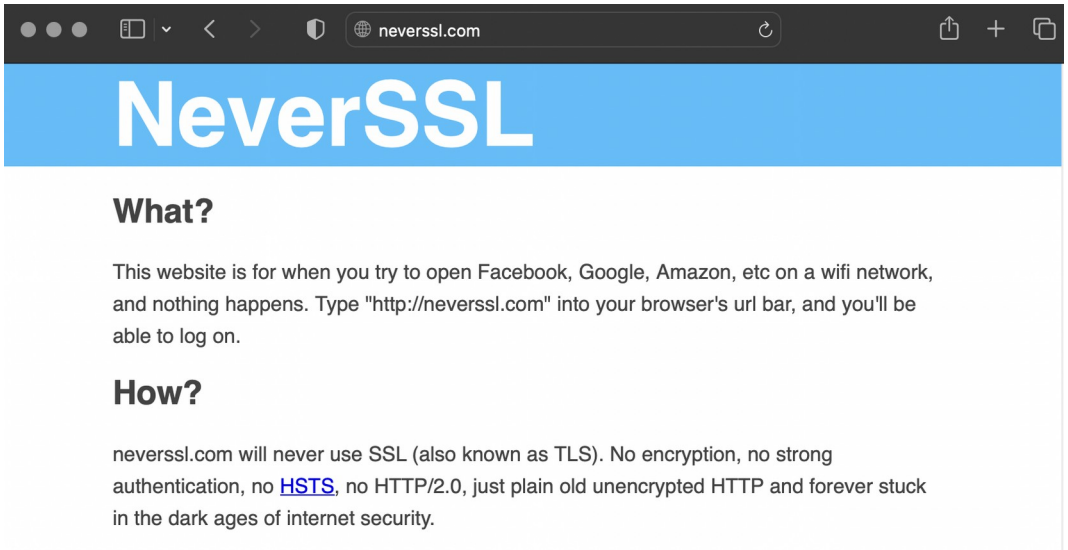
For this laboratory activity, select the network interface that your computer is using to connect to the Internet, which would typically be either the **Wi-Fi** or the **LAN** interfaces. If you aren't sure which one exactly, if you observe the sample image, you could see a line on the right of the interface name, which shows a simple line graph depending on the network traffic. The more jagged the line, doesn't necessarily mean it is the priority or default interface that you machine is using, but you can try choosing one of them for now and just change later on if its not the right one. The demo machine would use the **LAN** interface since that is its priority or default interface.

For this laboratory activity, we would explore some of the basic functionalities of Wireshark to help you get started. Note that if your computer is using IPv6 to communicate, you can disable it for now so that the output would be consistent to that of the laboratory activity guide. If you are using MacOSX, open **Network Preferences** then select the interface used to connect to the Internet, then choose **Advanced** then on the **TCP/IP** tab, look for the **Configure IPv6** option then select **Link-local only** or **Off** then select **OK** and then **Apply**. For Windows, open **Control Panel** then **Network and Internet** then **Network and Sharing Center**, afterwards select **Change adapter settings** then right-click on the interface used to connect to the Internet and select **Properties**, then untick the **Internet Protocol Version 6 (TCP/IPv6)** option then select **OK**.

Now then, to begin packet capture, select the interface for Wireshark. You can verify which interface you have selected by observing the Title Bar of Wireshark or the bottom bar as well. By default, you should also see some rows of data being appended in the table at the center of Wireshark, these are actual packets being captured that are going in and out of your network interface, similar to the image provided below:




The interface window shows the packets being captured, which should be the default, else you can always select the ***Capture>Start*** button from the menu or click the **Start capturing packets** ( or ) button. By selecting the ***Capture>Stop*** ( ) button from the menu or by clicking on the **Stop capturing packets** button, you can stop or pause the packet capture at anytime. But don't stop packet capture yet! Let's capture some interesting packets first. To do this, we would need to generate some network traffic, which we can do so using a web browser, so while the packet capture is running, enter the following URL: http://neverssl.com, as shown in the image provided below:

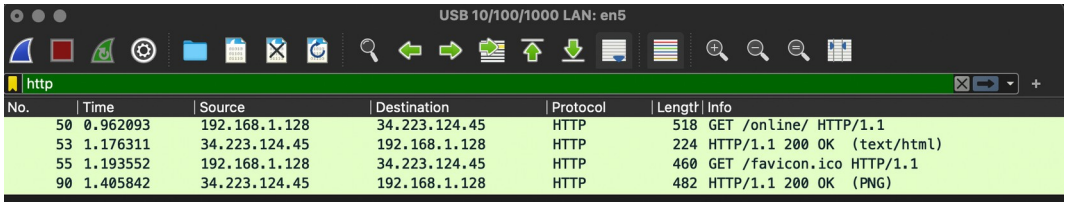


***NeverSSL*** is a website that is said to never use SSL or TLS, hence it would not have encryption, strong authentication, and such as is indicated in its webpage. Since it would not be implementing encryption, this would be a good test server for our laboratory activity.

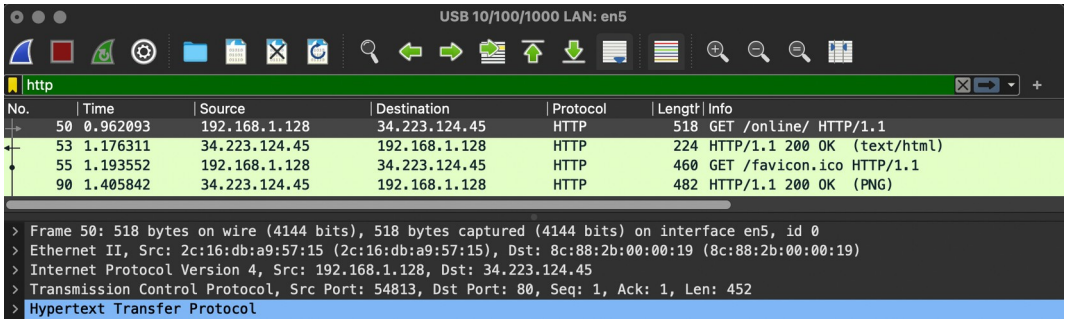
In order to display this page, your browser will contact the ***neverssl.com*** web server through the domain name and exchange HTTP messages with the server in order to download this page and render it on your browser. The messages containing these exchanges as well as all other data passing through your network interface would also be captured by Wireshark. After your browser has displayed the said page, stop the packet capture by selecting the **Stop capturing packets** button. You should now have the live packet captures which contains all the protocol messages exchanged between your computer and the server!

The HTTP message exchanges with the ***neverssl.com*** web server should appear somewhere in the listing of packets captured, but note that there would be many other types of packets displayed as well. Even though the only action you took was to open and download a web page, there were evidently many other protocols running on your computer that are unseen by the user.

Filter the packets by typing **http** into the **Display Filter Expression** textfield at the top portion of Wireshark. Find the HTTP GET message that was sent from your computer to the ***neverssl.com*** web server.




When you select the **HTTP GET** message (packet no. 50 in the sample image provided), the Ethernet frame, IP datagram, TCP segment, and HTTP application message header information would be displayed in the **Packet-header window**. Recall that the **HTTP GET** message that is sent to the ***neverssl.com*** web server is contained within a TCP segment, which is then contained or encapsulated in an IP datagram, which is once again encapsulated in an Ethernet frame.




Investigate the Ethernet frame and identify the following information: [1pt]

| | |
|---|---|
| Ethernet (MAC) Destination Address | CiscoMer_10:be:80 (ac:17:c8:10:be:80) |
| Ethernet (MAC) Source Address | IntelCor_a2:71:0f (f4:4e:e3:a2:71:0f) |

Investigate the IP Datagram and identify the following information: [2.5pts]

| | |
|---|---|
| Version | 4 |
| Header Length | 20 bytes (5) |
| Total Length | 482 |
| Internet Protocol (IP) Source Address | 10.33.197.125 |
| Internet Protocol (IP) Destination Address | 34.223.124.45 |

Investigate the TCP segment and identify the following information: [2.5pts]

| | |
|---|---|
| Source Port | 62573 |
| Destination Port | 80 |
| TCP Segment Len | 442 |
| Flags | 0x018 (PSH, ACK) |
| TCP Payload | 442 bytes |

Investigate the HTTP application data header and identify the following information: [3pts]

| | |
|---|---|
| Host | neverssl.com\r\n |
| Request Method | GET |
| Request URI | / |
| Request Version | HTTP/1.1 |
| User-Agent | Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik |
| Accept-Language | en-US,en;q=0.9\r\n |

What do you think is the purpose of encapsulation and why are we encapsulating so much information and at multiple layers at that, whenever we send data from one machine to another. Use the space provided below: [2pts]

Encapsulation organizes data to a form that can be transmitted over the network, with the information being sent through different layers prior to the destination. From my understanding, the purpose of encapsulation is to make it easier to identify what data are being transferred in each layer of the network. This is because by organizing the information through encapsulation, it becomes easier to differentiate where the data is from and what type of data they are. Furthermore, they are done in multiple layers to ensure that in case there is an error in one of the protocols, it is easier to identify the error and how far the data has reached.

What do you think is the role of each layer in the encapsulation process as you examined them. Do you think that each layer is performing something unique or are they redundant? Use the space provided below: [2pts]

Examining the layers, it is clear that each role has purpose. For the Ethernet frame, it identifies the hardware or MAC address of the end device, particularly of the device sending the data (source) and the device receiving the data (destination). Meanwhile, the IP Datagram identifies a host and route in the Internet for the devices to connect, while also carrying the information to be transferred. For the TCP segment, it exists to ensure that there is a reliable transmission of data from the source to destination, as well as a way to detect errors and corect them. Lastly, the HTTP is there to provide users the protocols needed to communicate in the network.

With all the tasks given to you for this Laboratory Activity, can you summarize your learnings and findings by providing a Conclusion. The Conclusion may discuss some realizations as you are doing the laboratory activity, analysis, and even reflection points and so on. Use the space provided below: [2pts]

Having completed the activity, it is clear that the transmission of data in a network is a complex process involving the exchange of data between layers, with each layer serving a unique purpose. Based on the information retrieved from each layer, I realized that the process would include needing to first get the address of the devices, finding a host to transfer the data, using a certain protocol for transmitting the data without errors, and of course to have a layer that serves as an interface for the client to transfer data between devices. By organizing data in such as way and through encapsulation, it allows the process to be easier to analyze as it becomes clearer where any potential problems may occur. It also provides an opportunity to find any vulnerabilities, as one can see how the data is transferred from start to end, with the layers serving as a guide to know at which step the transfer is at.

After completing the activity, don't forget to save and submit the filled up manual.

Have fun!