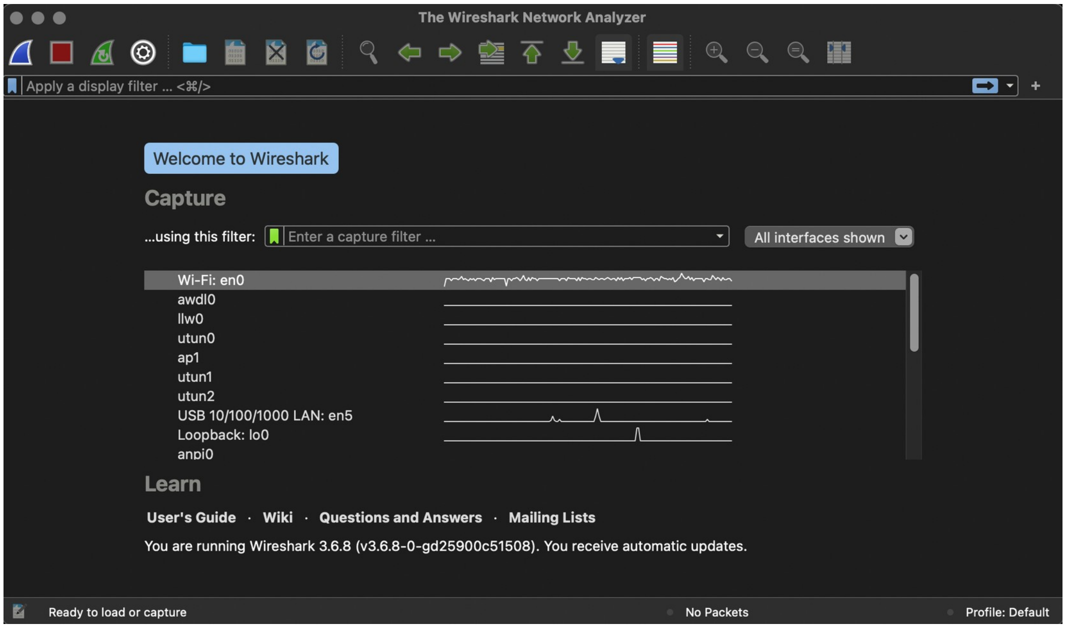
De La Salle University
College of Computer Studies
CSNETWK Laboratory Activity 1 - Investigating Encapsulation

Name: Kintanar, Shaylene and QUIRANTE, Julian   Section: S13

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:



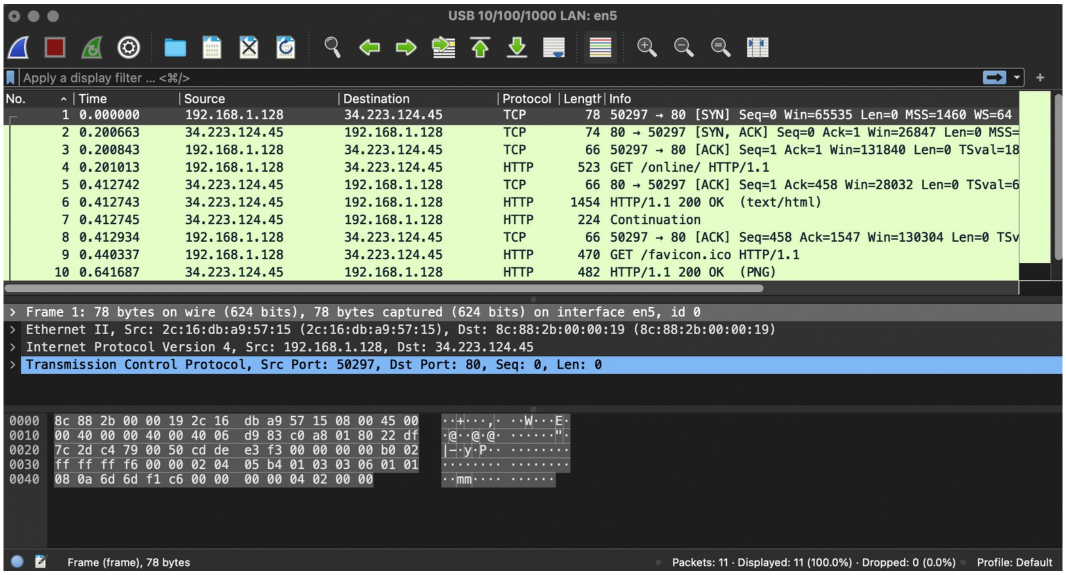
You are running Wireshark 3.6.8 (v3.6.8-0-gd25900c51508). You receive automatic updates.

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:



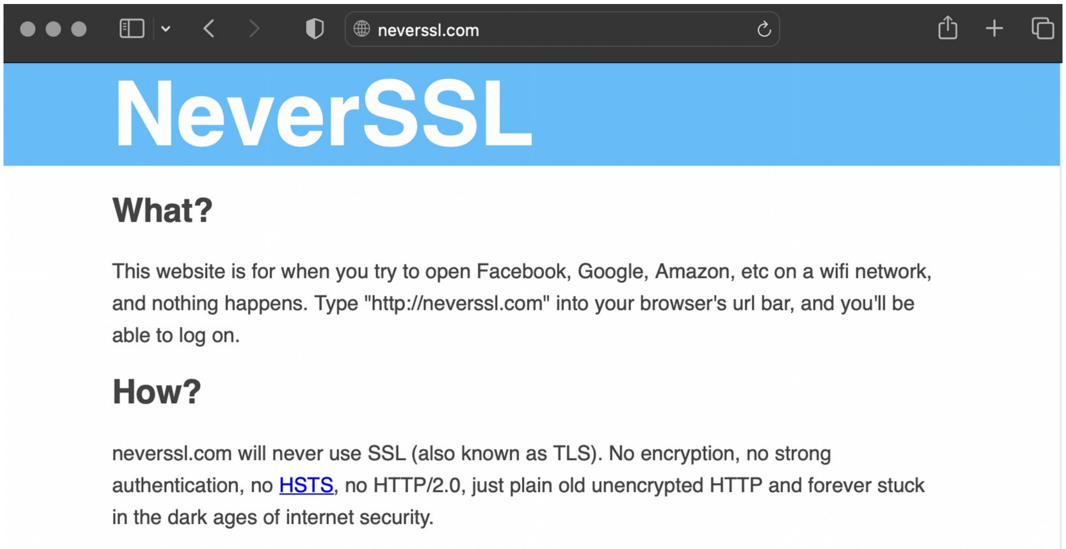
```
No.    Time       Source          Destination     Protocol Lengtł Info
     1 0.000000   192.168.1.128   34.223.124.45   TCP        78 50297 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
     2 0.200663   34.223.124.45   192.168.1.128   TCP        74 80 → 50297 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=
     3 0.200843   192.168.1.128   34.223.124.45   TCP        66 50297 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=18
     4 0.201013   192.168.1.128   34.223.124.45   HTTP      523 GET /online/ HTTP/1.1
     5 0.412742   34.223.124.45   192.168.1.128   TCP        66 80 → 50297 [ACK] Seq=1 Ack=458 Win=28032 Len=0 TSval=6
     6 0.412743   34.223.124.45   192.168.1.128   HTTP     1454 HTTP/1.1 200 OK  (text/html)
     7 0.412745   34.223.124.45   192.168.1.128   HTTP      224 Continuation
     8 0.412934   192.168.1.128   34.223.124.45   TCP        66 50297 → 80 [ACK] Seq=458 Ack=1547 Win=130304 Len=0 TSv
     9 0.440337   192.168.1.128   34.223.124.45   HTTP      470 GET /favicon.ico HTTP/1.1
    10 0.641687   34.223.124.45   192.168.1.128   HTTP      482 HTTP/1.1 200 OK  (PNG)
```

```
> Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface en5, id 0
> Ethernet II, Src: 2c:16:db:a9:57:15 (2c:16:db:a9:57:15), Dst: 8c:88:2b:00:00:19 (8c:88:2b:00:00:19)
> Internet Protocol Version 4, Src: 192.168.1.128, Dst: 34.223.124.45
> Transmission Control Protocol, Src Port: 50297, Dst Port: 80, Seq: 0, Len: 0
```

```
0000  8c 88 2b 00 00 19 2c 16  db a9 57 15 08 00 45 00   ··+···,· ··W···E·
0010  00 40 00 00 40 00 40 06  d9 83 c0 a8 01 80 22 df   ·@··@·@· ······"·
0020  7c 2d c4 79 00 50 cd de  e3 f3 00 00 00 00 b0 02   |-·y·P·· ········
0030  ff ff ff f6 00 00 02 04  05 b4 01 03 03 06 01 01   ········ ········
0040  08 0a 6d 6d f1 c6 00 00  00 00 04 02 00 00         ··mm···· ······
```

```
Frame (frame), 78 bytes                    Packets: 11 · Displayed: 11 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

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

What?

This website is for when you try to open Facebook, Google, Amazon, etc on a wifi network, and nothing happens. Type "http://neverssl.com" into your browser's url bar, and you'll be able to log on.

How?

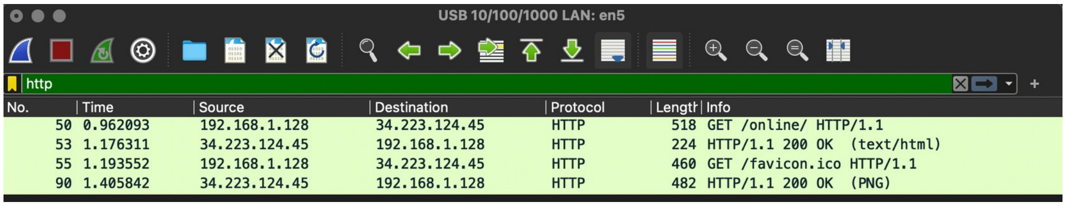
neverssl.com will never use SSL (also known as TLS). No encryption, no strong authentication, no HSTS, no HTTP/2.0, just plain old unencrypted HTTP and forever stuck in the dark ages of internet security.

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!

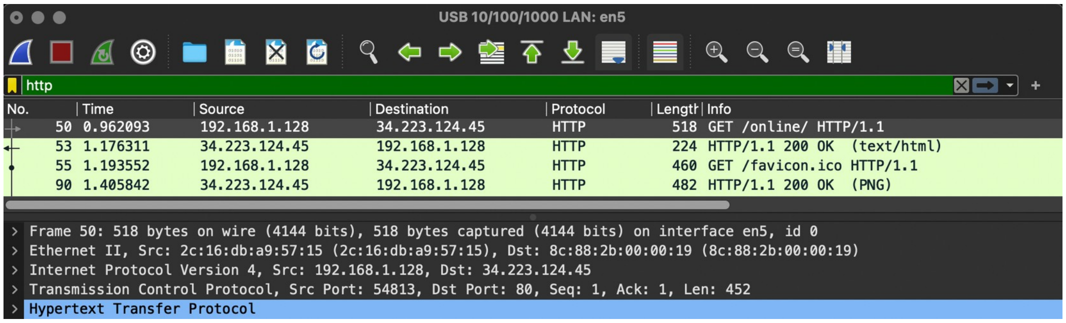
The HTTP message exchanges with the ***neverssl.com*** web server should appear somewhere in the listing of packets captured, but note that there would be many other types of packets displayed as well. Even though the only action you took was to open and download a web page, there were evidently many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



```
USB 10/100/1000 LAN: en5
http
No.  Time      Source         Destination    Protocol  Length Info
  50 0.962093  192.168.1.128  34.223.124.45  HTTP      518 GET /online/ HTTP/1.1
  53 1.176311  34.223.124.45  192.168.1.128  HTTP      224 HTTP/1.1 200 OK  (text/html)
  55 1.193552  192.168.1.128  34.223.124.45  HTTP      460 GET /favicon.ico HTTP/1.1
  90 1.405842  34.223.124.45  192.168.1.128  HTTP      482 HTTP/1.1 200 OK  (PNG)
```

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.



```
USB 10/100/1000 LAN: en5
http
No.  Time      Source         Destination    Protocol  Length Info
  50 0.962093  192.168.1.128  34.223.124.45  HTTP      518 GET /online/ HTTP/1.1
  53 1.176311  34.223.124.45  192.168.1.128  HTTP      224 HTTP/1.1 200 OK  (text/html)
  55 1.193552  192.168.1.128  34.223.124.45  HTTP      460 GET /favicon.ico HTTP/1.1
  90 1.405842  34.223.124.45  192.168.1.128  HTTP      482 HTTP/1.1 200 OK  (PNG)

> Frame 50: 518 bytes on wire (4144 bits), 518 bytes captured (4144 bits) on interface en5, id 0
> Ethernet II, Src: 2c:16:db:a9:57:15 (2c:16:db:a9:57:15), Dst: 8c:88:2b:00:00:19 (8c:88:2b:00:00:19)
> Internet Protocol Version 4, Src: 192.168.1.128, Dst: 34.223.124.45
> Transmission Control Protocol, Src Port: 54813, Dst Port: 80, Seq: 1, Ack: 1, Len: 452
> Hypertext Transfer Protocol
```

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

Ethernet (MAC) Destination Address	98:00:6a:1a:32:42
Ethernet (MAC) Source Address	34:6f:24:c3:5d:9b

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

Version	4
Header Length	20 bytes
Total Length	467
Internet Protocol (IP) Source Address	192.168.254.111
Internet Protocol (IP) Destination Address	34.223.124.45

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

Source Port	54314
Destination Port	80
TCP Segment Len	427
Flags	0x0018
TCP Payload	TCP payload (427 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]

The purpose of encapsulation is to ensure that data transmission is performed properly. Data encapsulation can include error detection, sequencing of data, congestion control, routing data, and etc. Sending so much information without layers would be complex and difficult to manage. Having layers means that they can each focus on a specific task. A device only has to understand its own layer and the layer below it for effective communication. Layers also provide a common language for different devices, allowing them to communicate seamelessly.

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]

The application layer focuses on user applications like browsers or email clients, it prepares the data to travel through the network by adding headers. The transport layer is basically for data delivery where data is broken down into segments or datagrams to ensure they arrive at their destination in the proper order. The network layer addresses the encapsulated data in the layer (packets) with source and destination IP addresses which lets them travel across different networks. The data link layer deals with the transmission of data over a network like Wi-Fi. It adds MAC addresses to identify devices on the network and error detection for the next layer. The

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]

The laboratory activity solidified our understanding of encapsulation and the OSI layers by applying concepts learned during class time to practice. Through this activity, we learned more about how each layer has its own function for preparing the data. Each of these layers' functions are essential for seamless transmission between devices. Further, we learned that preparing data is more than transforming data and placing them in packets but also attaching additional information to follow network and communication protocols which are essential for eficient and secure transfer of data.

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

Have fun!