Lab 02 – HTTPS Decryption

Author: Raymond Ng
Course Number/Section: IS 3413-006
Date: September 6, 2022

Introduction

The purpose of this lab is to allow the user to experiment with HTTPS decryption. In this lab the user will use Secure Socket Layer (SSL) to decrypt Transport Layer Security (TLS) packets via Wireshark. Further, this lab will serve as introductory experience and allow the user to experiment with the use of the Firefox DevOps browser.

Process

Per the instructions, I began the lab by downloading Firefox DevOps from https://www.mozilla.org/en-US/firefox/developer. After the downloaded completed, I opened the browser and typed in `about:config` in the search bar to access the configuration editor preferences. Next, I created a new configuration with `NSS_ALLOW_SSLKEYLOG=1` and set the Boolean value true (depicted in Figure 1).

Figure 1: NSS_ALLOW_SSLKEYLOG=1 configuration setting.

Breakpoint 1: No issues to report during this stage of the lab.

Breakpoint 2: Per the instructions, I opened my Task Manager to verify if any other browsers were running. No other browsers were running except for Firefox Developer Edition (as depicted in Figure 2).

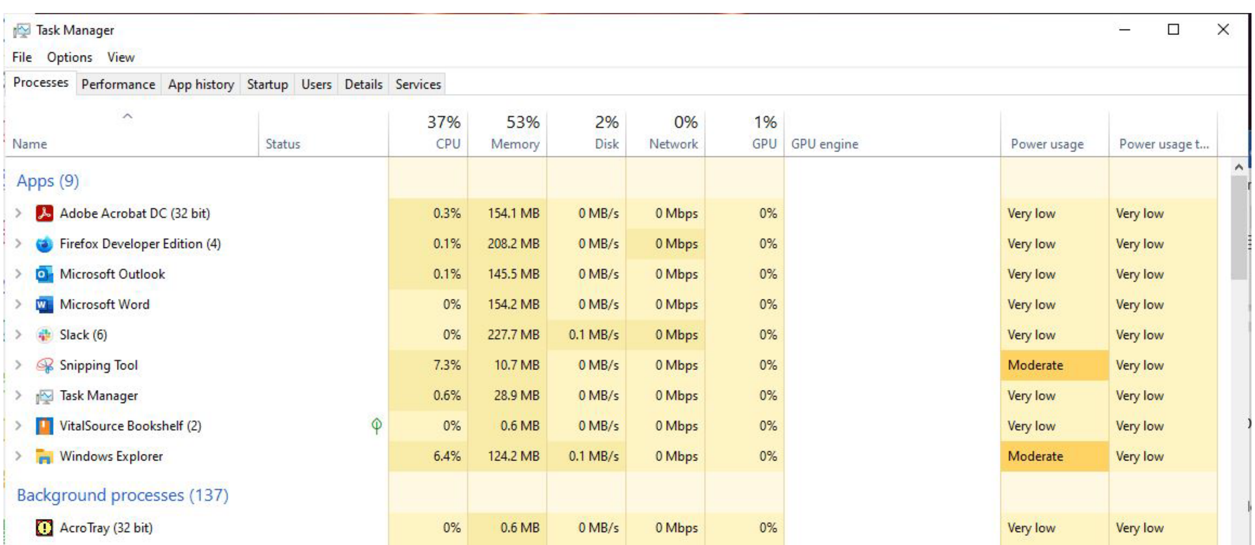

Figure 2:Windows Task Manager Processes Tab

Next, I opened PowerShell to set the variable `SSLKEYLOGFILE`. First, I changed my directory to my Documents folder by executing `cd Documents` in PowerShell. Then, I set the variable by executing `setx SSLKEYLOGFILE "$(get-location)\ssl.log`. (Figure 3)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\rayng> cd Documents
PS C:\Users\rayng\Documents> setx SSLKEYLOGFILE "$(get-location)\ssl.log"

SUCCESS: Specified value was saved.
PS C:\Users\rayng\Documents>
```

Figure 3:Setting the SSLKEYLOGFILE environmental variable in PowerShell.

To verify the that the environmental variable was set, I opened up a new PowerShell window and executed Get-ChildItem ENV: | findstr SSLKEYLOGFILE. (Figure 4)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\rayng> cd Documents
PS C:\Users\rayng\Documents> Get-ChildItem ENV: | findstr SSLKEYLOGFILE
SSLKEYLOGFILE                  C:\Users\rayng\Documents\ssl.log
PS C:\Users\rayng\Documents>
```

Figure 4:Verified the SSLKEYLOGFILE setting.

Breakpoint 3: In this part of the lab, I opened Wireshark and began a packet capture. Next, I opened a Firefox DevOps browser and went to https://www.google.com, and typed in "*wireshark*" in the search bar of the website and stopped the packet capture in Wireshark. I saved my `pcap` file to the same place my `ssl.log` was located on my computer. (Figure 5)

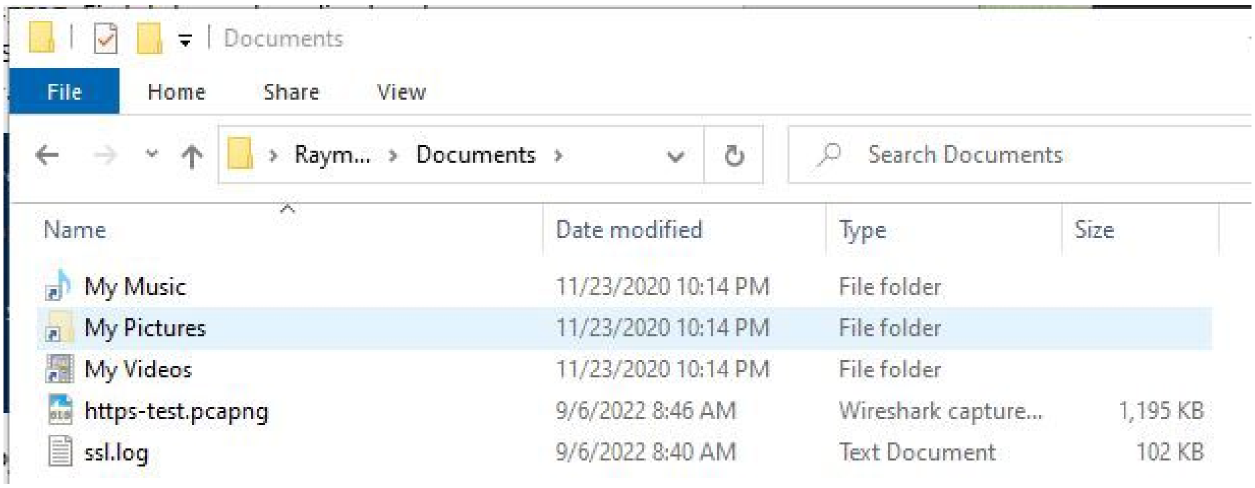

Figure 5:The ssl.log and pcap file with timestamps.

Breakpoint 4: In this part of the lab, I opened my packet capture file (`pcap` file) that I saved from the previous step and began examining the packet capture file. In the Protocols column, I observed several Protocols to include `UDP`, `TLSv1.3`, `TLSv1.2`, `TCP`, `SNMP`, `QUIC`, `OSCP`, `HTTP3`, `HTTPS`, `HTTP2`, `HTTP`, `DoH`, `DNS`, and `ARP`. In the Info column, I saw a lot of additional information as they relate to each packet, including `GET`, which I know is a method/command found in the `HTTP` request lines. Moreover, I noticed `HTTP` responses statuses like `HTTP/1.1` (`HTTP` version number) and Status code and text like `200 OK`. I assessed the Info column depicted the requests or the responses as the pertain to each packet.

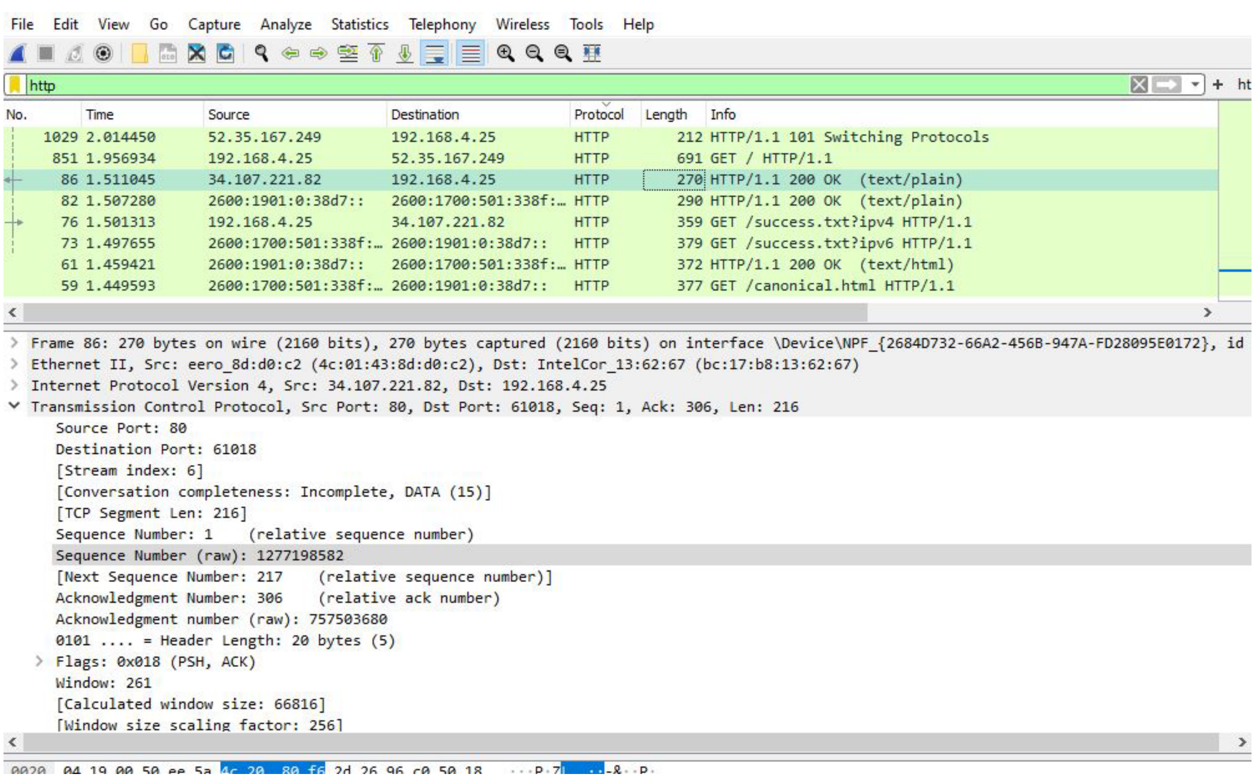

Figure 6: pcap file with an HTTP filter

After filtering my packet capture file by `HTTP` in Wireshark, I concluded that both the client's browser and the web server were using `HTTP/1.1`, which I learned was a standardized protocol [1]. (Figure 6)

Continued observation of the packet capture, filtered by `HTTP` in Wireshark, I assessed that the client machine's (web browser) IP address was `34.107.221.82`, port `80`.

From observing the same packet that I used to determine the IP and port of the client machine, I assessed that the host machine's (web server) IP was `192.168.4.25`, on port `61018`.

Further observation of each packet, I noticed some packet were associated with the Online Certificate Status Protocol (OCSP) in the Protocol column. From techtarget.com, I learned OCSP specifies the syntax for communication between the server (contains the certificate status) and the client application (informed of the status). Moreover, OCSP enables real-time status checks on security certificates and is essential to the extended validation of SSL certificates. An example from *techtarget.com* indicates, when a user establishes a `HTTPS` connection with a web server, the browser typically performs an OCSP check with the certificate authority (`CA`) that issued the SSL certificate to ensure that it was not revoked; sometimes leads to short delays in the SSL handshake [2]. Furthermore, OCSP protocols' IP addresses appear to be in IPv6 format.

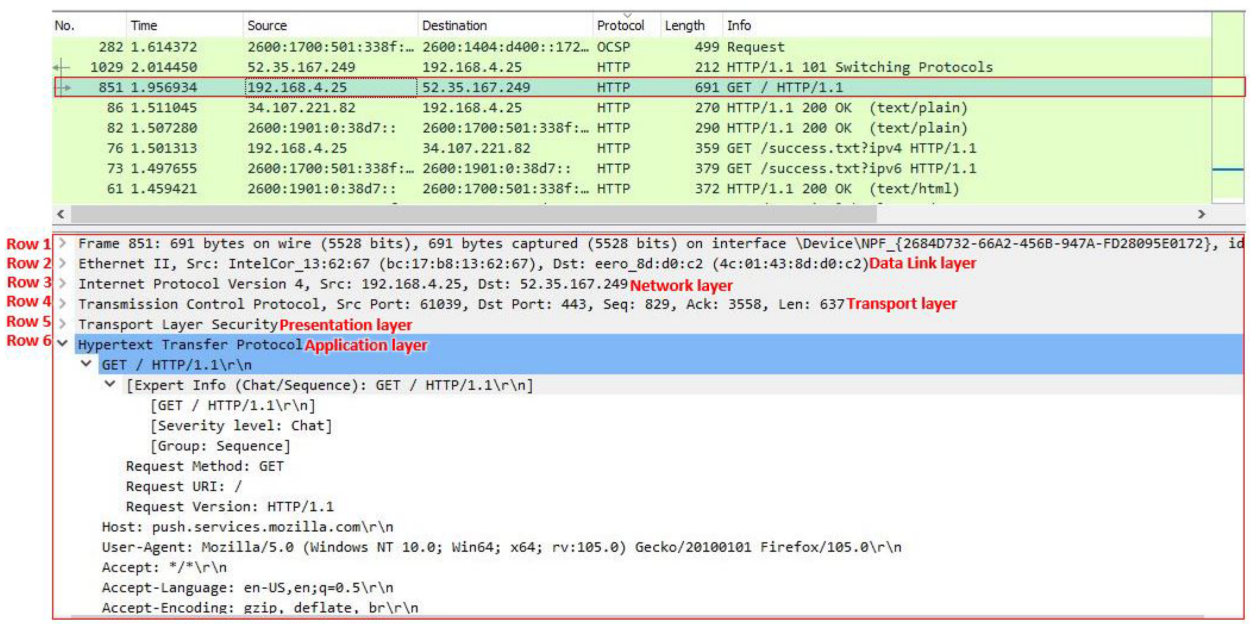

Figure 7: pcap file with one entry selected with the packet details pane.

Examining the Packet Details pane from one of the selected packets in the Packets List frame, I can determine the following (Figure 7):

- In **Row 1**, beginning with `Frame 851:`, provides the packet summary info. When I expanded I saw a description of the network interface Wi-Fi—the only network card connected to the internet—and the date-time stamp of the Arrival Time.

- In **Row 2**, beginning with `Ethernet II`, in relation to the OSI model, is the **Data Link** layer. I observed a couple of few couple of MAC addresses that looked familiar. I use Eero products to create a Wi-Fi mesh to extend the Wi-Fi signal in various parts of my house and recognized those MAC addresses from the Eero app that used to regulate the Eero Wi-Fi devices (Figure 8).

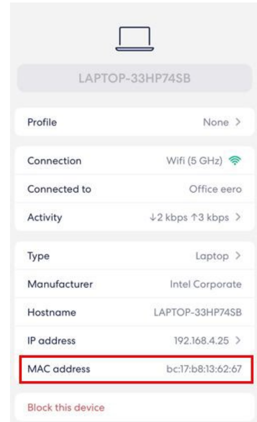

Figure 8: Screenshot of Eero mobile application.

- In **Row 3**, beginning with `Internet Protocol Version 4`, in relation to the OSI model, is the **Network** layer. It is depicting the traffic being routed from the host and client in IPv4 format.

- In **Row 4**, beginning with `Transmission Control Protocol`, in relation to the OSI model, is the **Transport** layer. From what I know from the previous module, it's the transmission of data segments between points, in this case from host and client ports.

- In Row 5, beginning with `Transportation Security Layer` (`TLS`), in relation to the OSI model, is assessed to be either the Session or the **Presentation** layers. There is some controversy online as to which layer TLS actually belongs to. From what I can glean from reading about TLS is that its essentially a secured line of communication between client and server where the data travels in encrypted format [3]. Thus, I assess that TLS is better associated with the **Presentation** layer.

- In **Row 6**, beginning with the `Hypertext Transfer Protocol` (`HTTP`), in relation to the OSI model, is assessed to be associated with the **Application** layer. When I expanded this row in the Packets Detail pane, I could interpret the HTTP request using the method/command `GET`. Further, I could interpret it was using HTTP version 1.1 in this row, as it was depicted in the HTTP request.

Breakpoint 5: Below is a screenshot of my environmental variable window before I deleted the `SSLKEYLOGFILE` entry from my operating system (Figure 9).

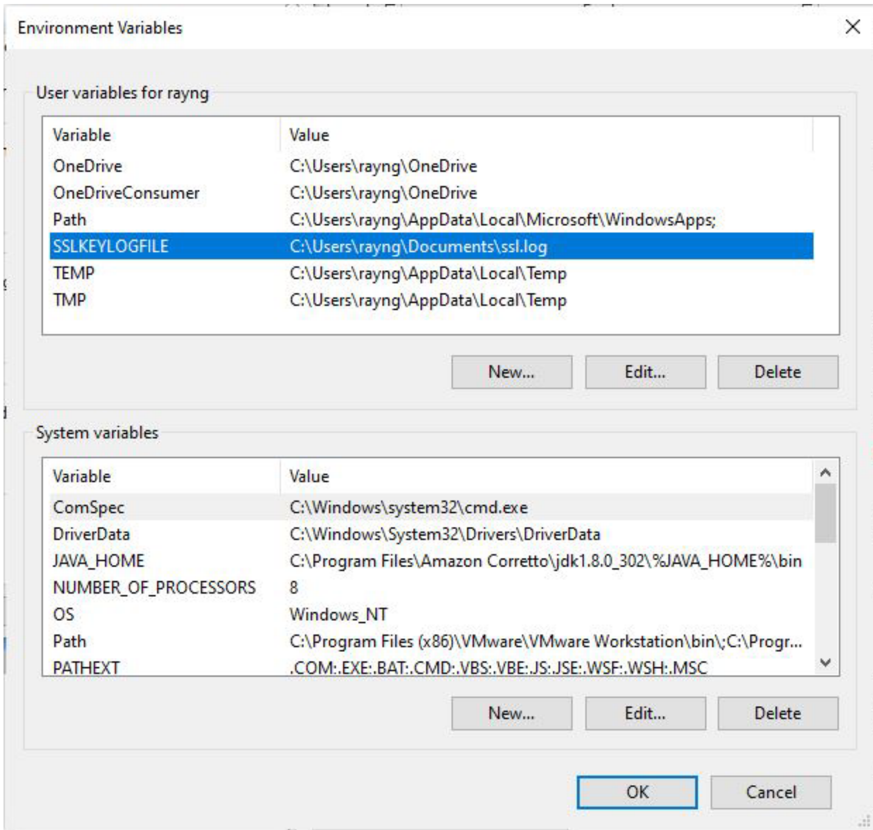

Figure 9: Deleting the SSLKEYLOGFILE environment variable form the operating system.

Limitations/Conclusion

The lab seemed to be fairly easy for the novice user like myself. I could not assess any limitation as everything was done on a live, host machine versus on a controlled environment, like on a VM. From this lab I learned how to interpret `HTTP` responses and requests. Moreover, I learned about `TLS` and `SSL` on the client side from the experimenting on this lab and researching via the Internet. The biggest takeaway was learning how to interpret the packets via Wireshark, and importing an `SSL` log file into Wireshark to analyze the unencrypted traffic.

References

[1] mdn [Online]. "Evolution of HTTP", August 22, 2022. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP [Accessed: 06-Sep-2022]

[2] Zola [Online]. "OCSP (Online Certififcate Status Protocol)", December 2021. Available: https://www.techtarget.com/searchsecurity/definition/OCSP [Accessed: 07-Sep-2022]

[3] Agrawal, Enterprise Engineering Content & Experience Services Blog [Online]. "All you need to know about TLSv1.2". March 4, 2019. Available: https://blogs.oracle.com/ee-ces/post/all-you-need-to-know-about-tlsv12 [Accessed: 07-Sep-2022]

Collaboration

Most of the lab was executed independently. I did post an inquiry on September 6, 2022 in our class Slack module-02 channel: https://is-3413-fall-2022.slack.com/archives/C03U235MHPH/p1662490033376359 to clarify one of the questions on the lab.