Lab *03* – Network Sniffing

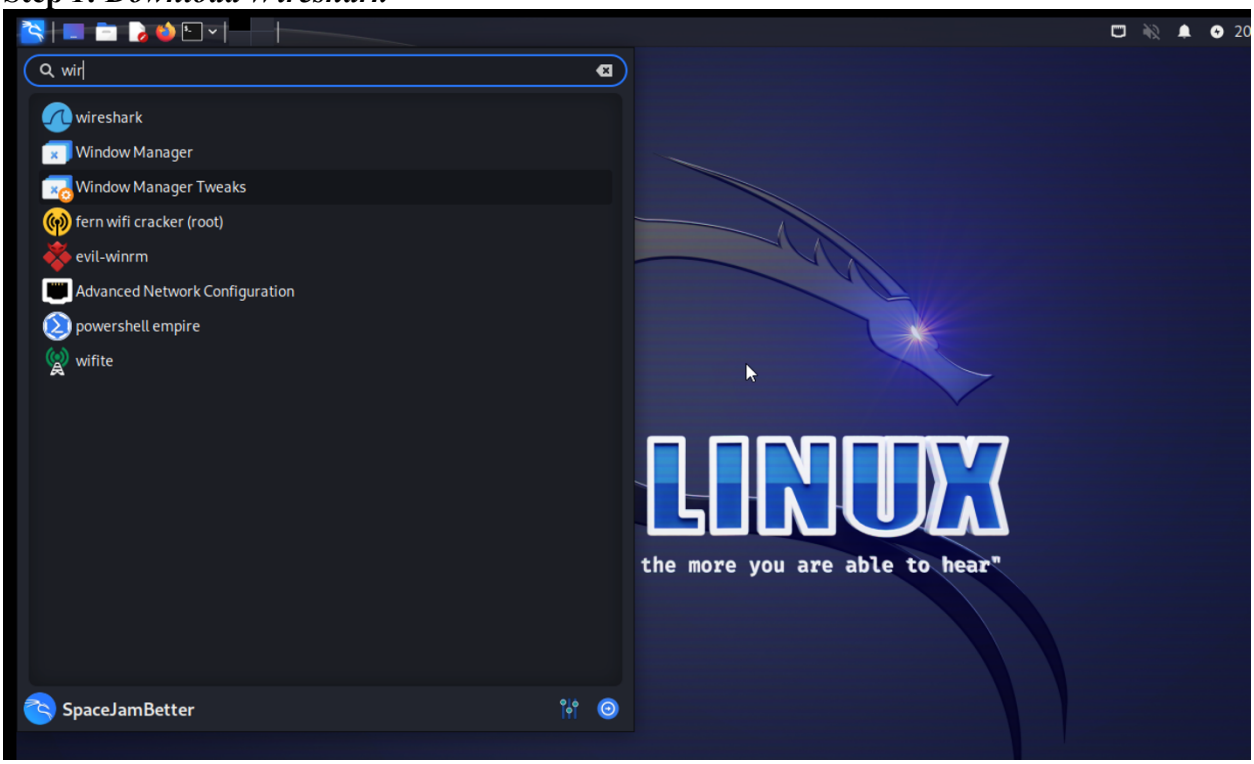
David Murillo Santiago
Professor Pugh
IS-3513
26 October 2023

Introduction

In this lab, I will learn the fundamentals of network packet analysis through Wireshark. I will acquire essential skills in capturing, filtering, and dissecting network packets, which are critical for understanding network traffic and ensuring security.

Process

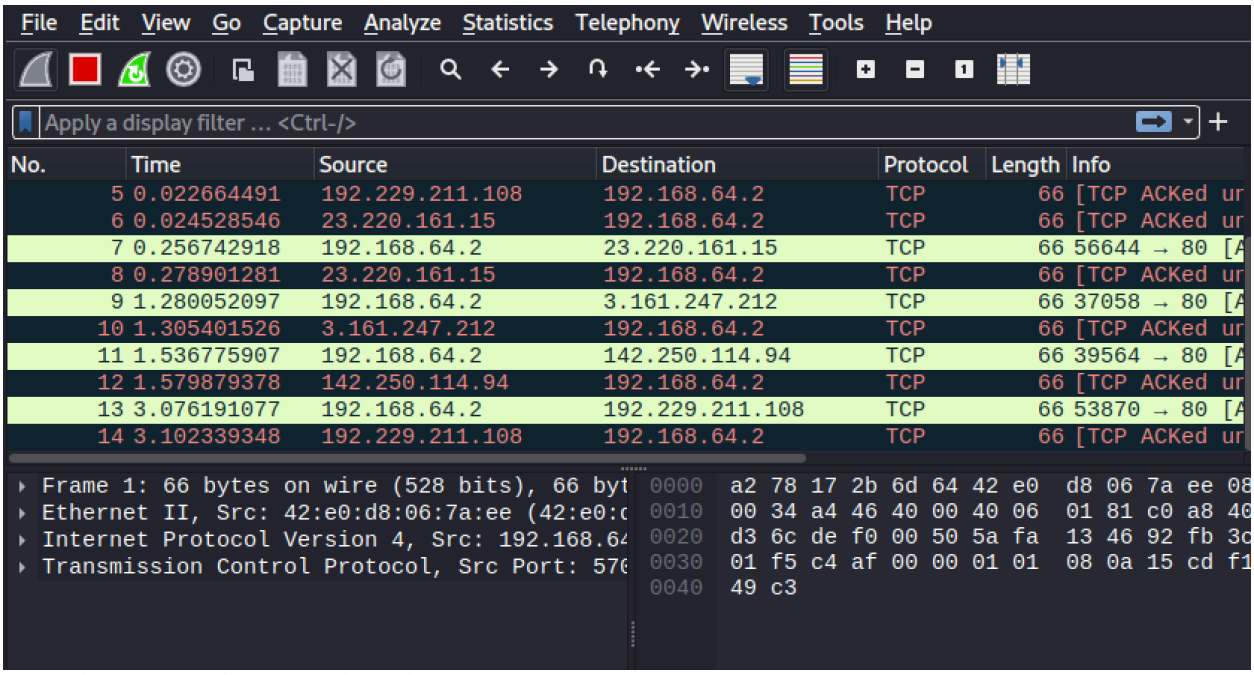
Step *1: Download Wireshark*



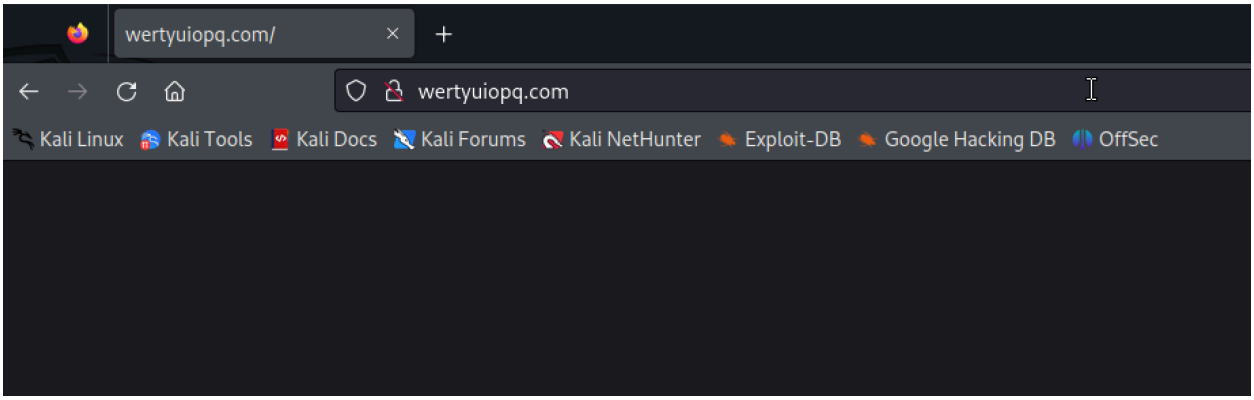
To begin, I needed to ensure that I had Wireshark. Because I am using a Kali Linux, Wireshark already comes downloaded with my machine.

Step *2: Packet Capture*

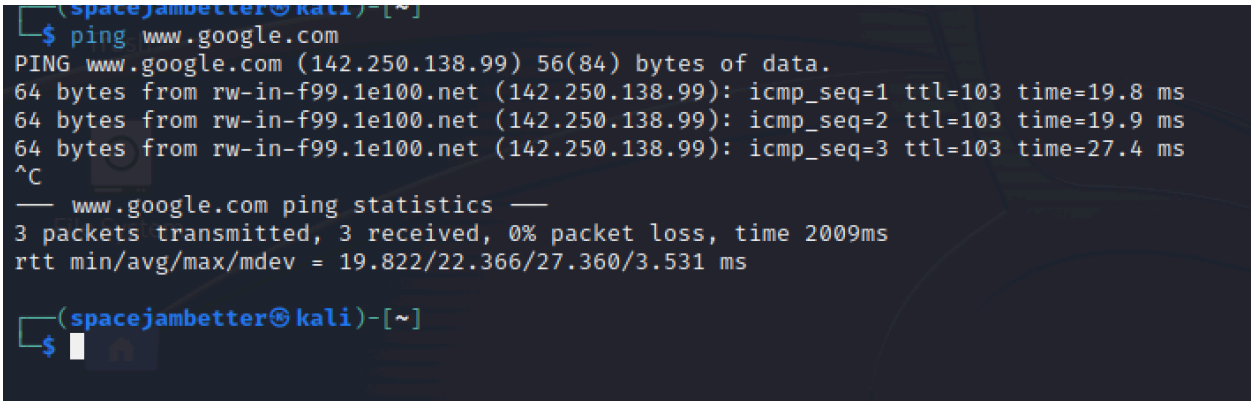
In this step, I began capturing packets on Wireshark and I searched for different websites.



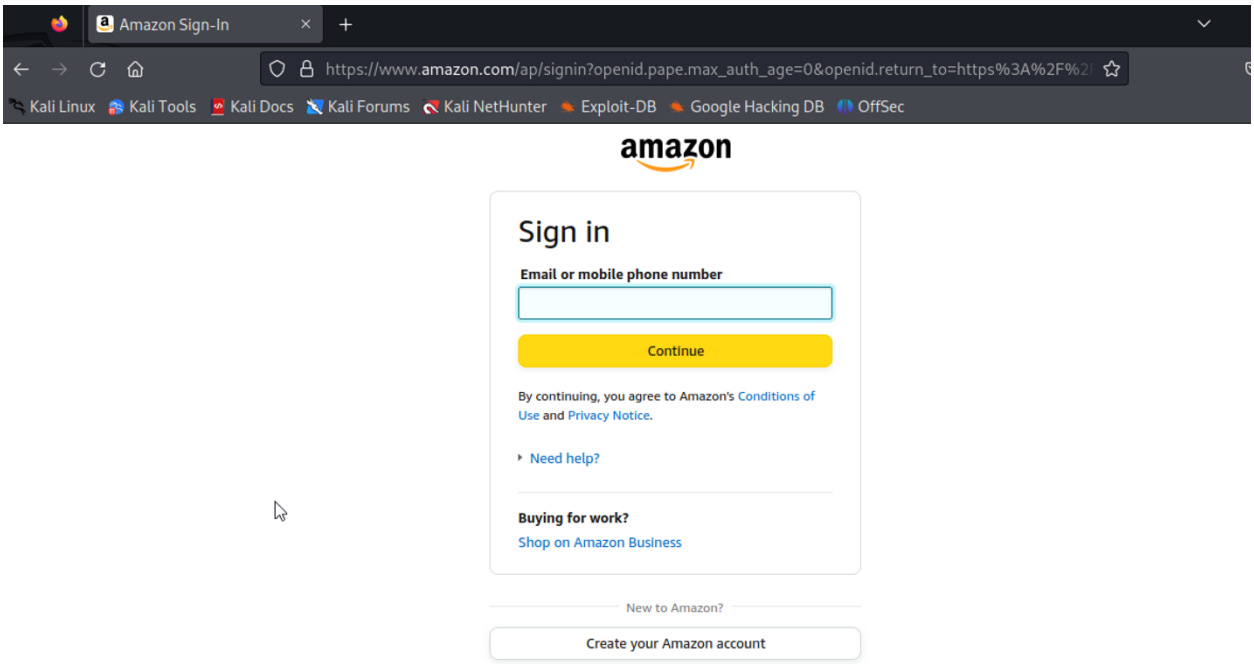
Screenshot revealing that I began the packet capture.



Next, I searched up a random website that does not exist. I searched up http://wertyuiopq.com. The search did not take me anywhere as the link was made up, therefore I was left on a blank, dark screen.



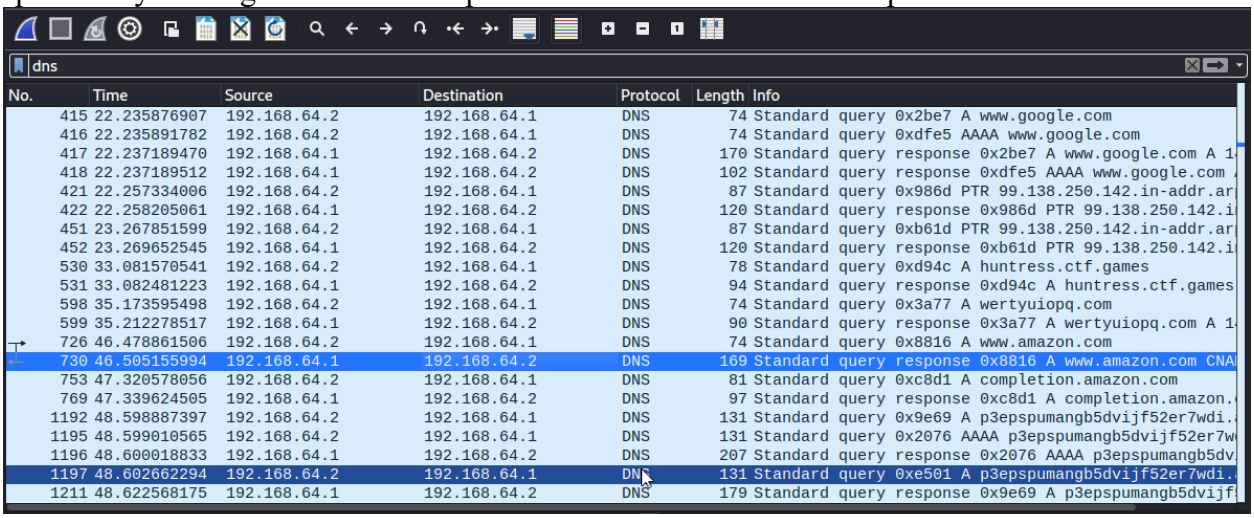
Next, I used the terminal to ping the google server. By pinging google, I found that their servers are responsive. Because I am using a Linux terminal, I needed to manually stop pinging google, so I entered "control + c" which ended the ping.



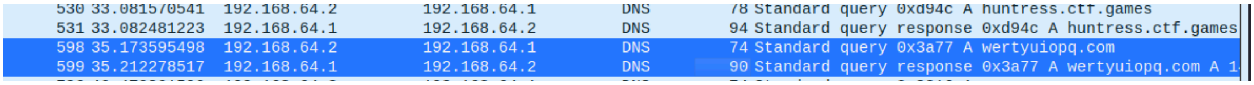
Next, I went on the Amazon website and clicked on "sign in." After this, I ended the packet capture.

Step *2: DNS packet analysis*

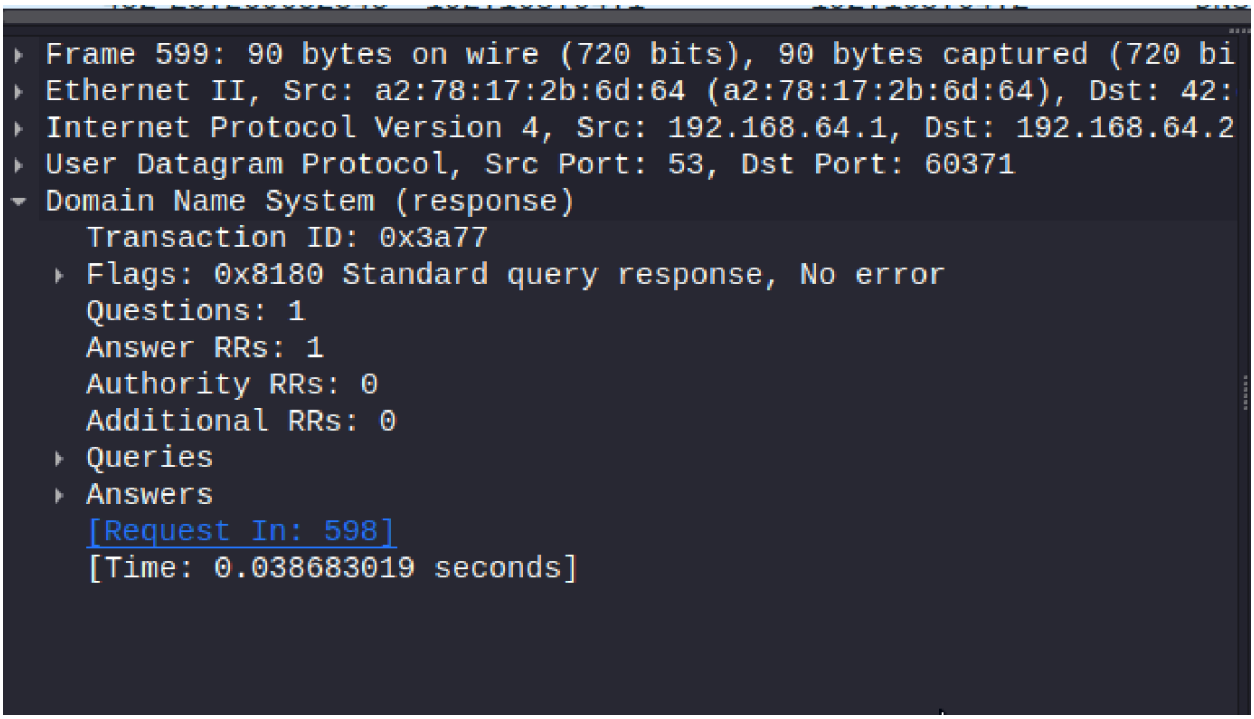
In this step I applied a filter on Wireshark that only displays the captured DNS packets. I was specifically looking for the DNS response to the fake website I attempted to search.



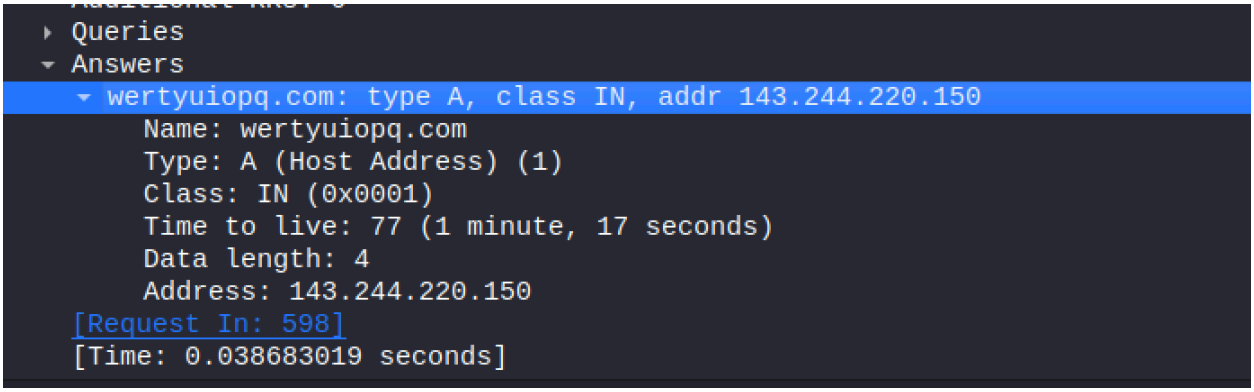
I entered DNS next to the Display Filter to only view DNS packets. This screenshot reveals all the captured DNS packets which includes the google, amazon, and fake website searches I made during the traffic capture.



Screenshot of the DNS query and response from my fake website search.



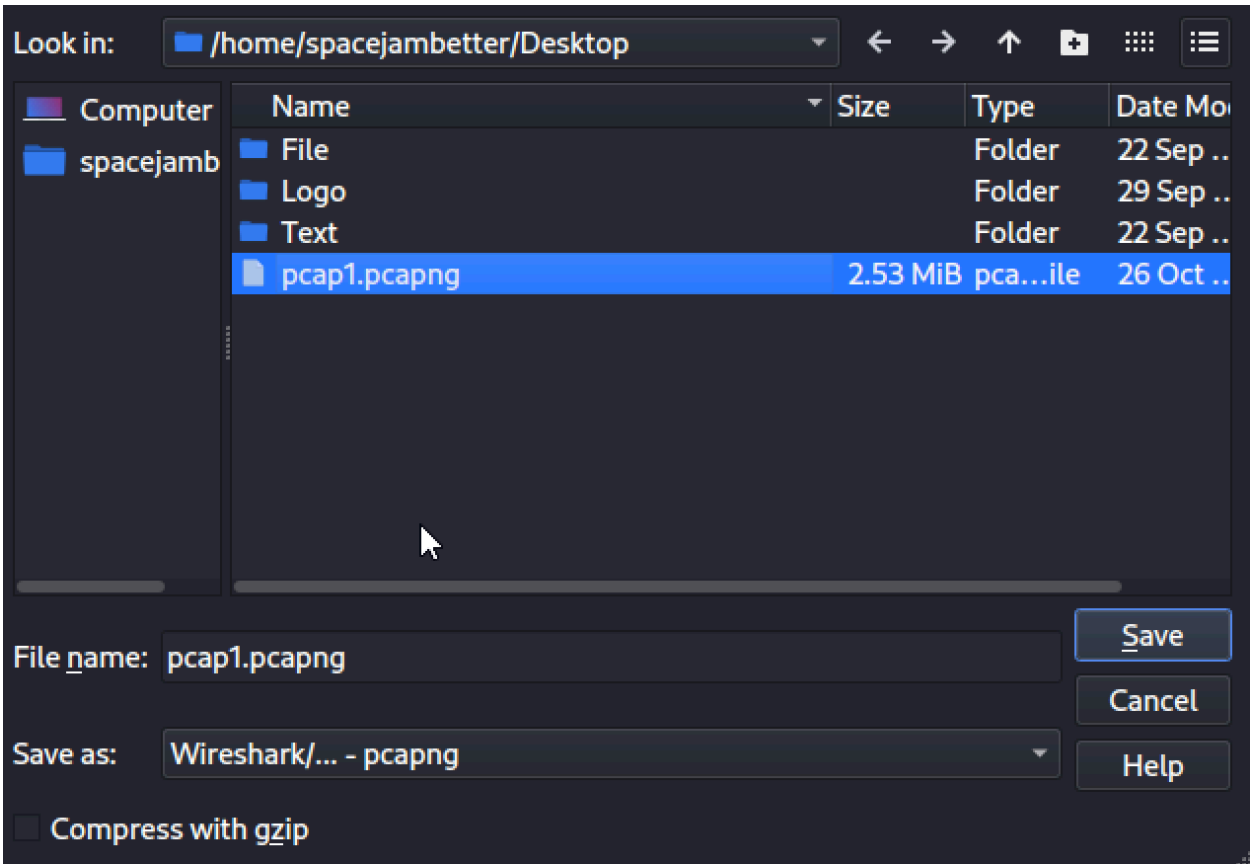
Next, I investigated the DNS response and found that there were no errors. This is odd because, if the website I searched is fake, the DNS response should include a "NXDOMAIN" or a "No such name."



Next, I investigated further and found that the "fake" website I had searched for was in fact real. The highlighted portion of the screenshot reveal the link, type, class, and address of the site.

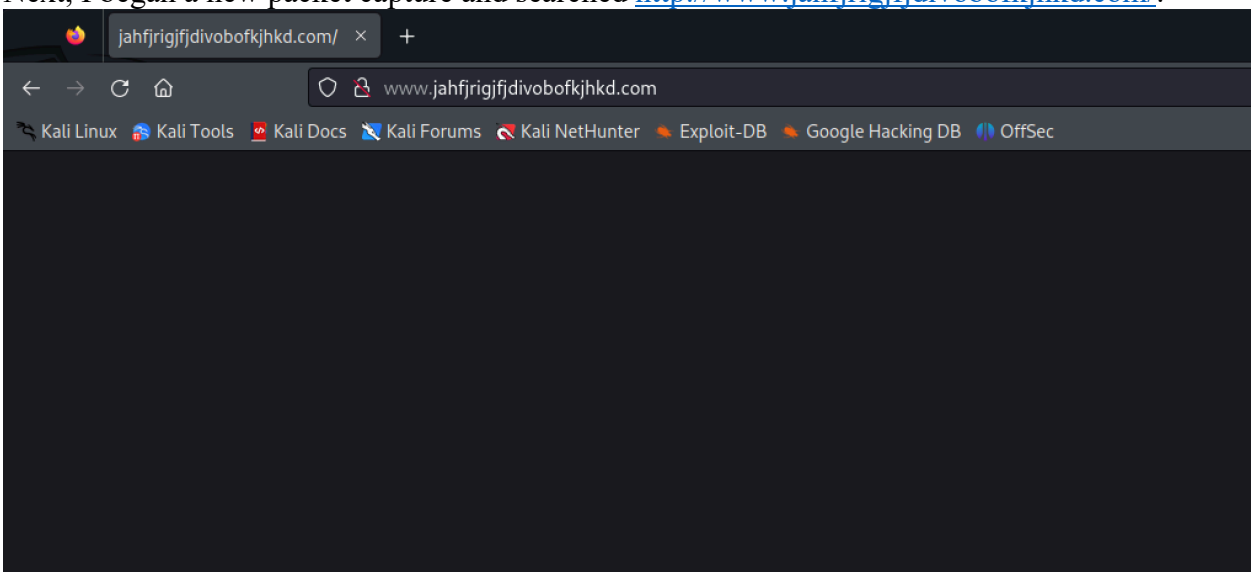
Once I realized the "fake" website was real, I decided to use the fake link provided in the instructions to guarantee that the website is fake.

First, I saved the packet capture to ensure the integrity of the remaining data which I must analyze.

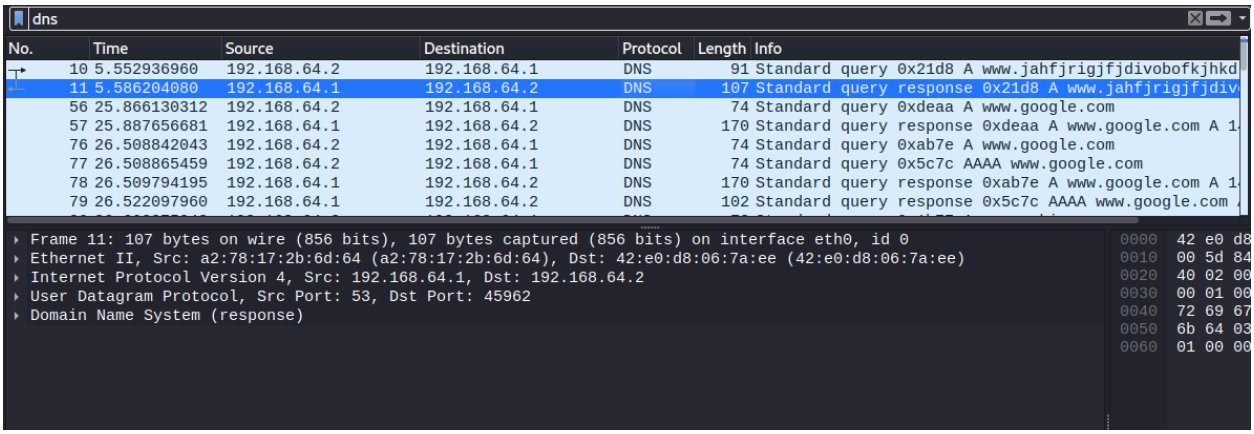


Screenshot revealing that I saved the packet capture as "pcap1" on my Desktop.

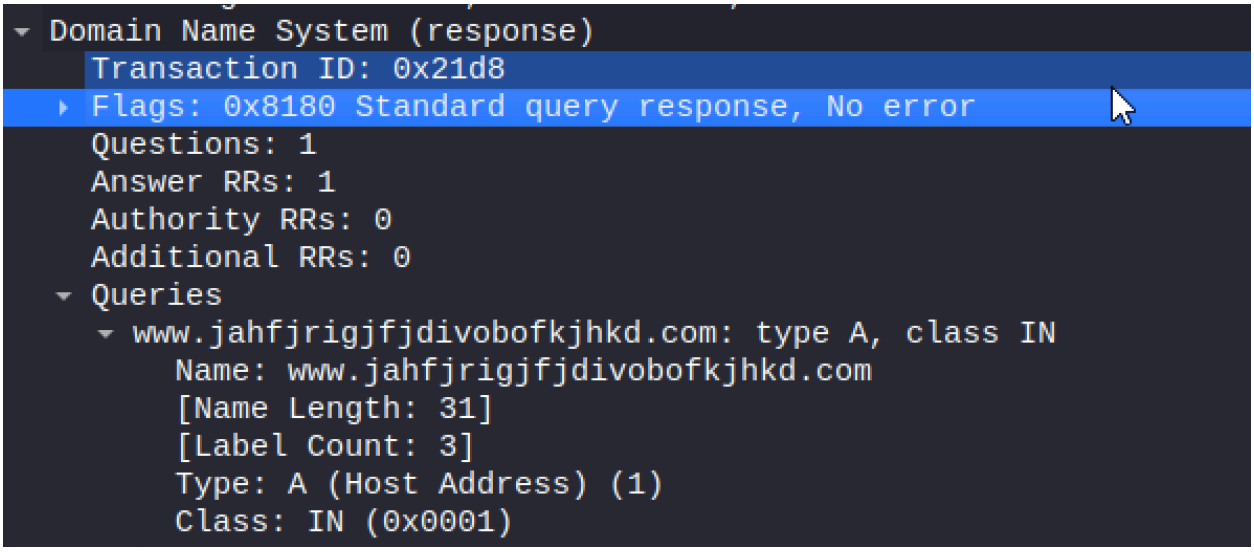
Next, I began a new packet capture and searched http://www.jahfjrigjfjdivobofkjhkd.com/.



Screenshot of me searching for the new, provided link.

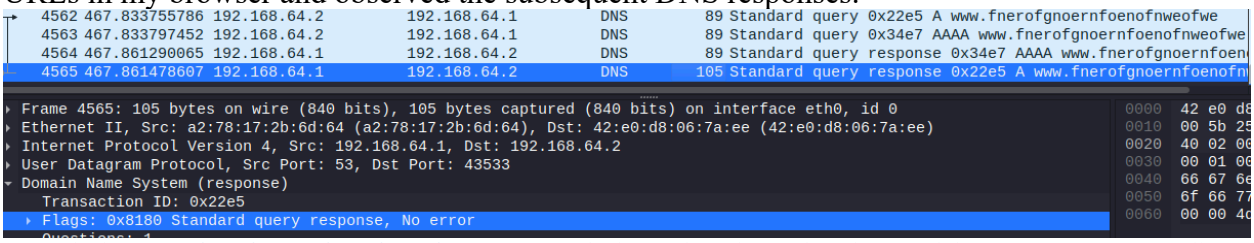


Screenshot revealing the new DNS packets that were captured.

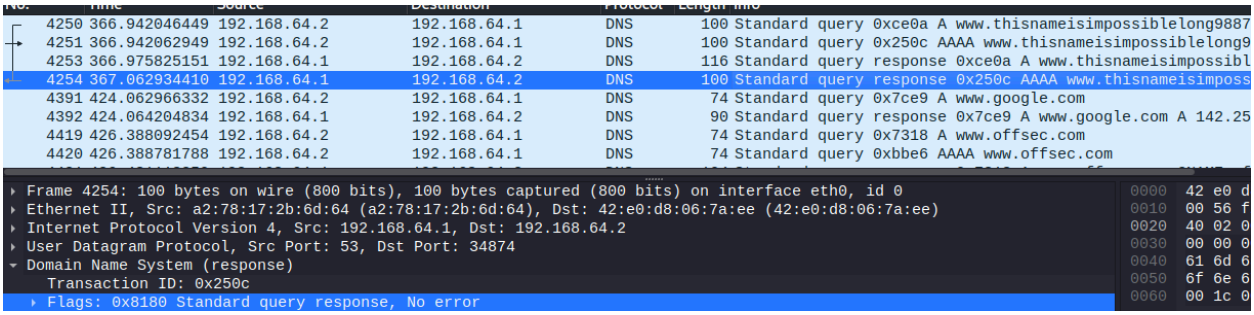


Once again, the DNS response was "No error."

To understand the meaning of "No error" I used the following website by BlueCat Networks: "The top four DNS response codes and what they mean" https://bluecatnetworks.com/blog/the-top-four-dns-response-codes-and-what-they-mean/
According to the site, "NOERROR" means that the DNS query was successfully completed, meaning that the DNS query for a domain name resolved to an IP address without any errors. One reason that the DNS response is "No error" instead of "no such name" is because my machine is configured to handle DNS queries for non-existent domains differently, possibly redirecting or modifying the DNS responses to provide alternative results or search pages rather than returning a standard "No such name" response. To test this I entered very fictious and long URLs in my browser and observed the subsequent DNS responses.

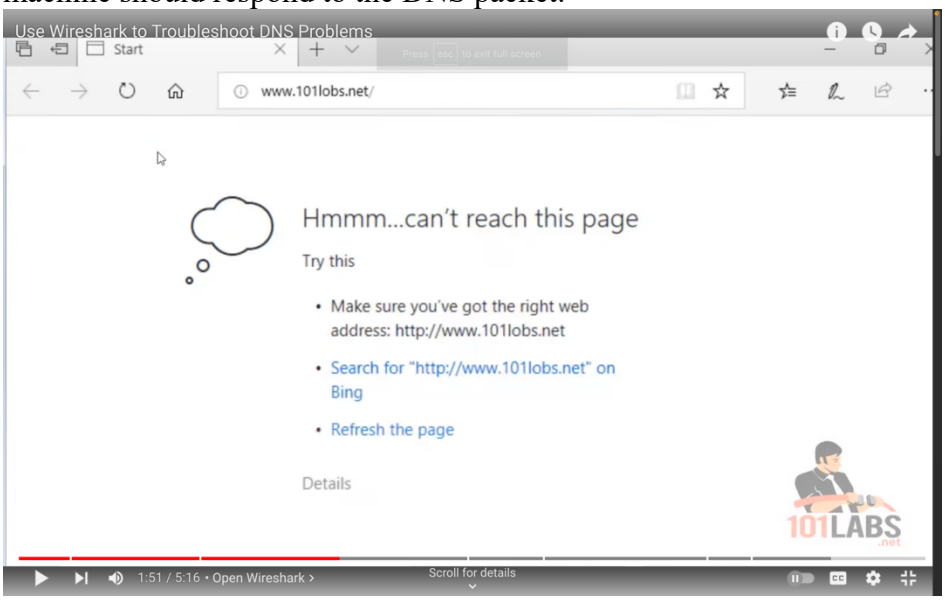


I searched up www.fnerofgnoernfoenofnweofwe.net, a site which I made up by mashing keys, and the DNS response was still "No error."

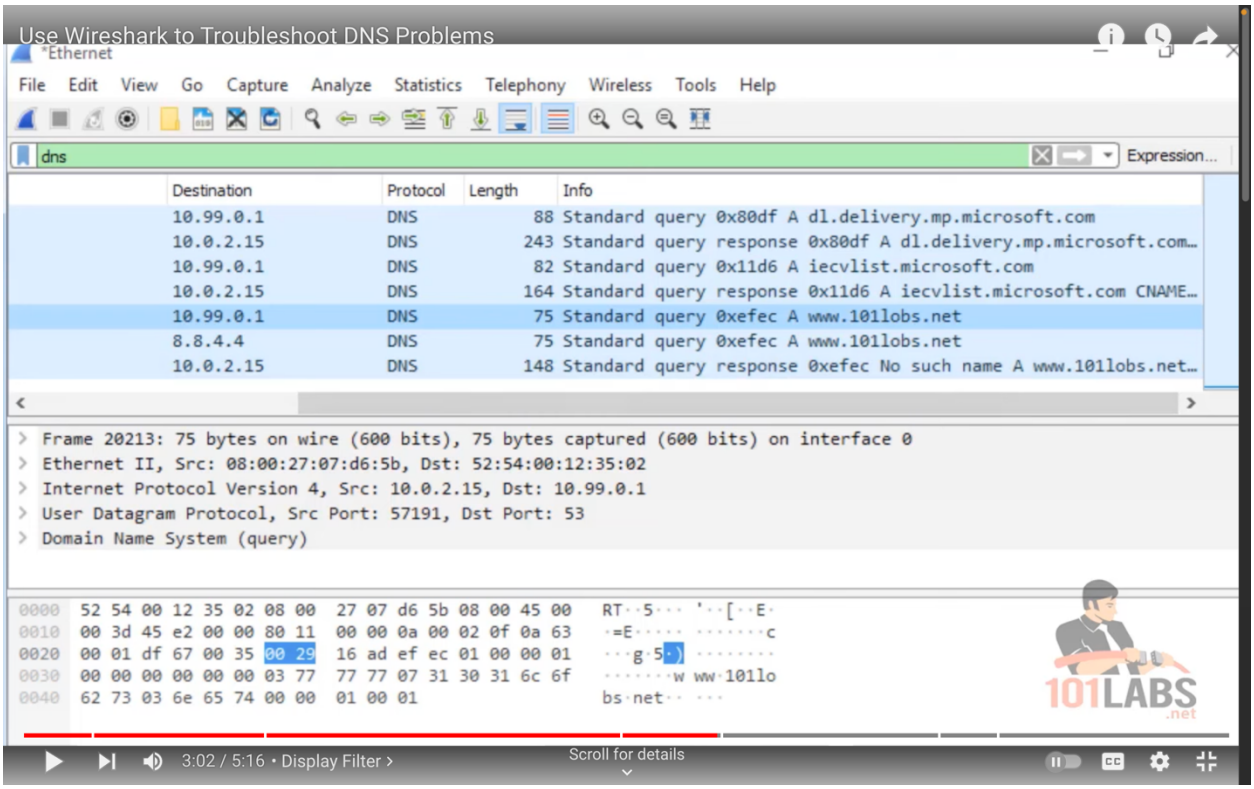


Next, I entered another fake link www.thisnameisimpossiblelong988776544563.net and the DNS response was once again "No error."

Finally, I watched the following YouTube video by Paul Browning: https://www.youtube.com/watch?v=vJsOteThRYA. This video gave me a reference for how my machine should respond to the DNS packet.

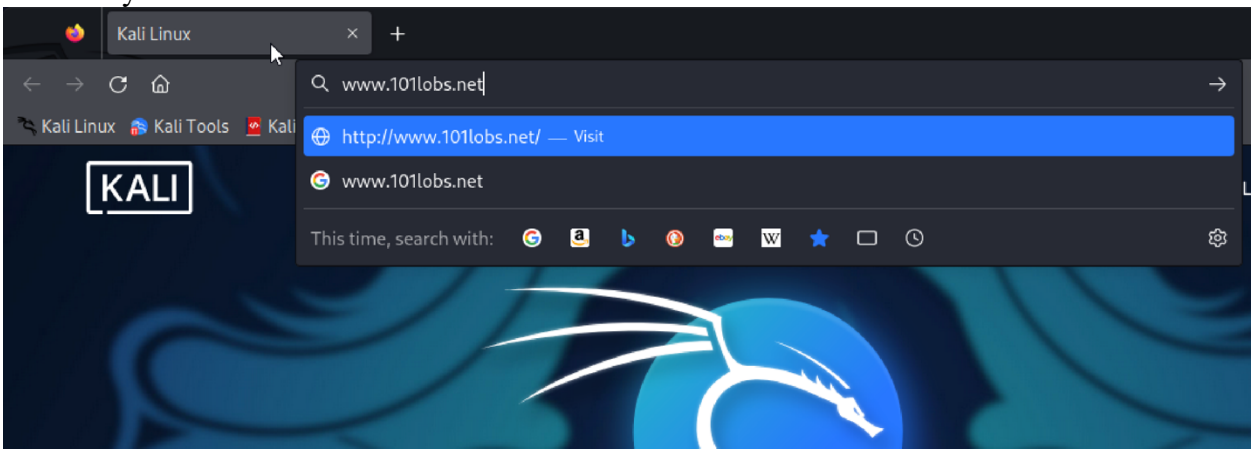


In the video, Browning searches www.101lobs.net/

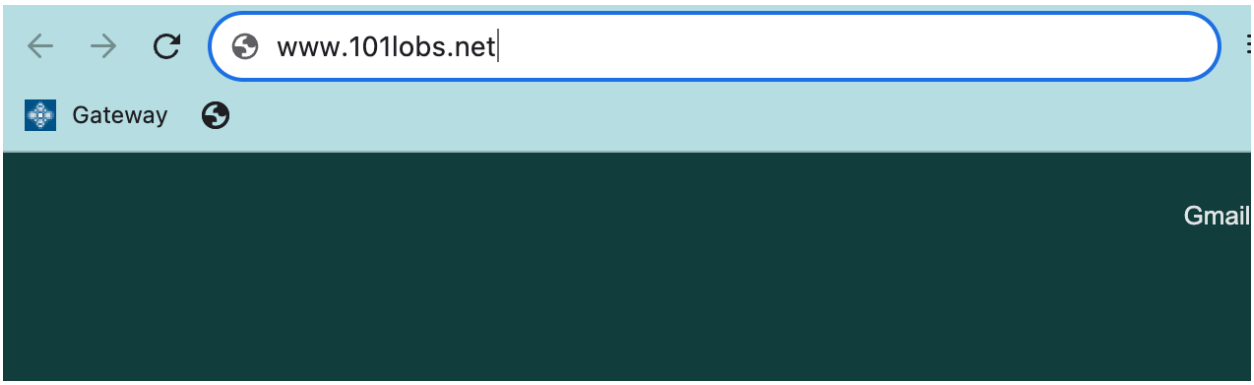


And as shown on the last packet, Browning receives a DNS response of "No such name."

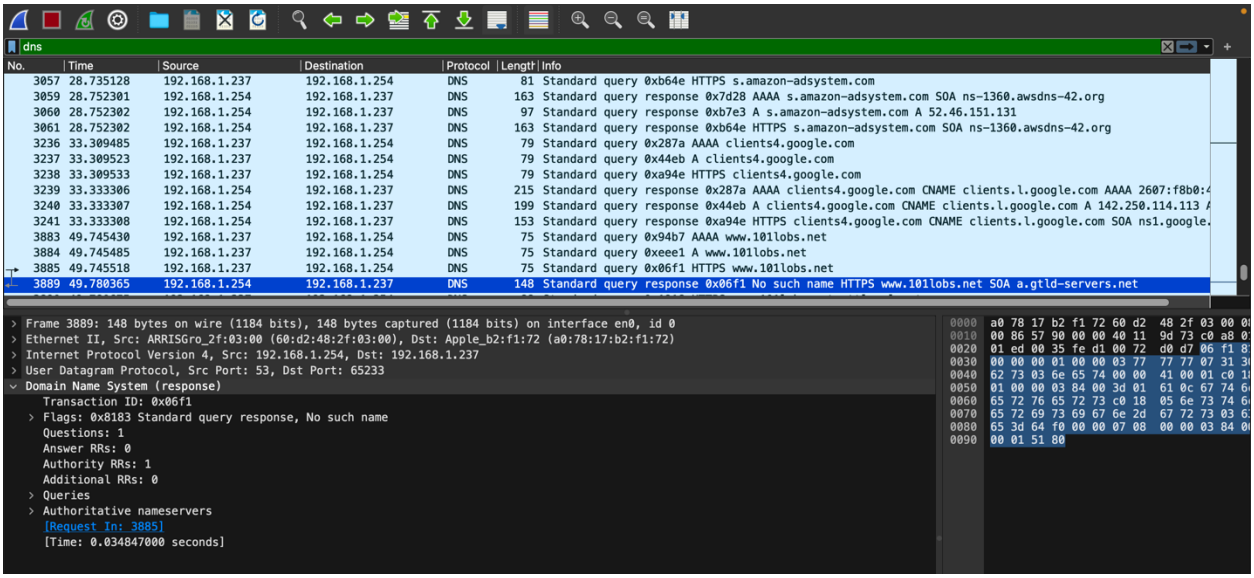
I attempted this process on both my Linux machine and my Mac to see if they would react differently.



I searched www.101lobs.net on my Linux machine.



I then searched for the same site on my Mac.

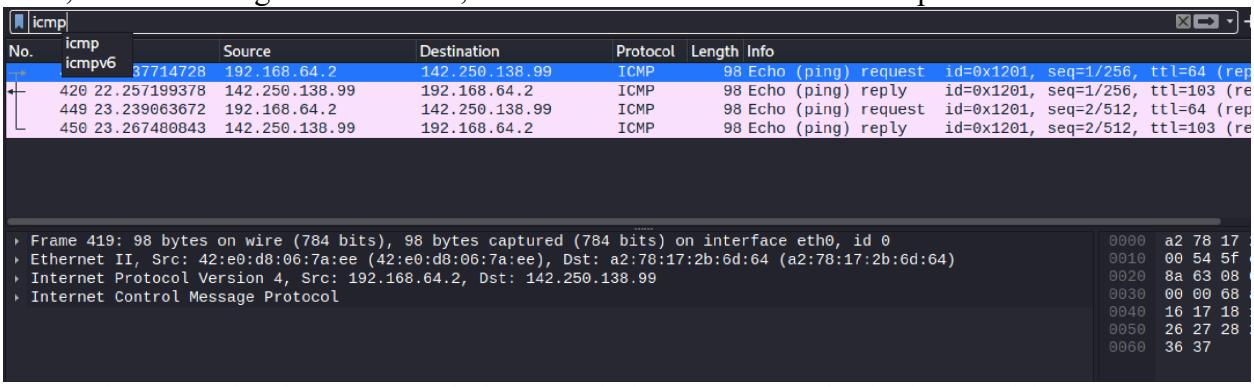


And finally, I got a "No such name" DNS response on Wireshark.

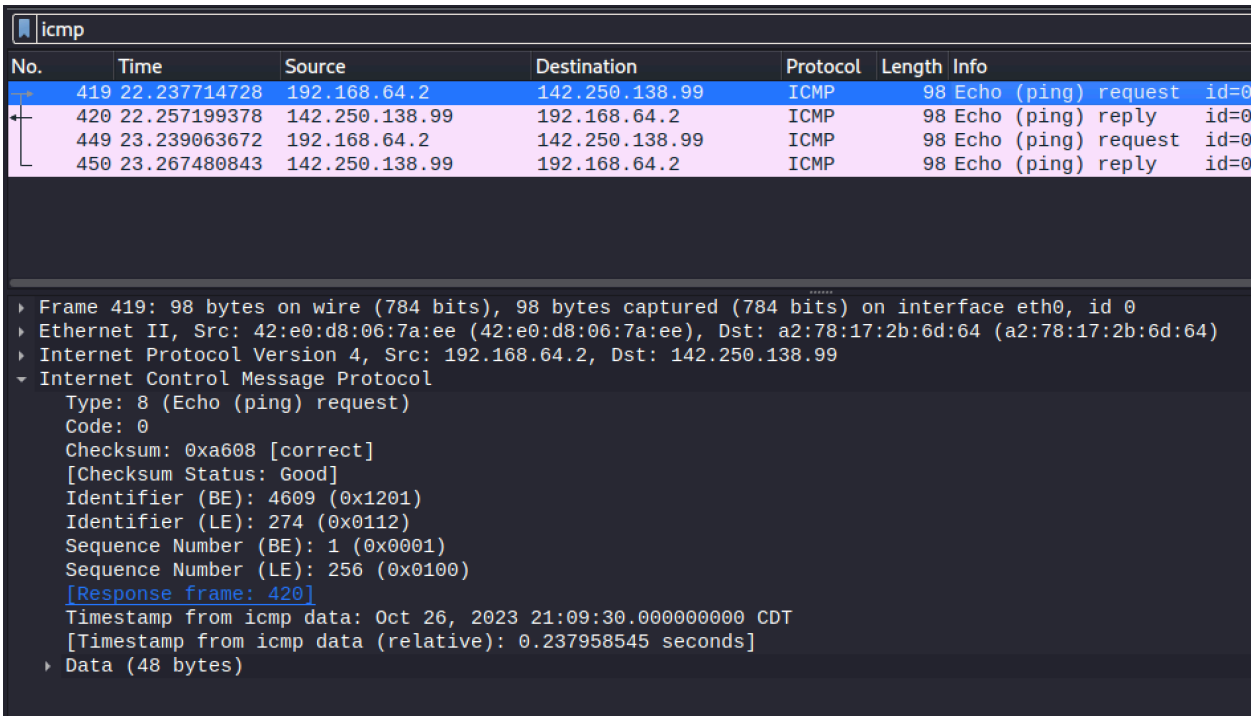
Because my two machines respond differently, it is likely that they are using separate DNS resolvers or DNS configuration settings, which can result in varying DNS response behaviors when querying the same domain.

Step *3: ICMP packet analysis*

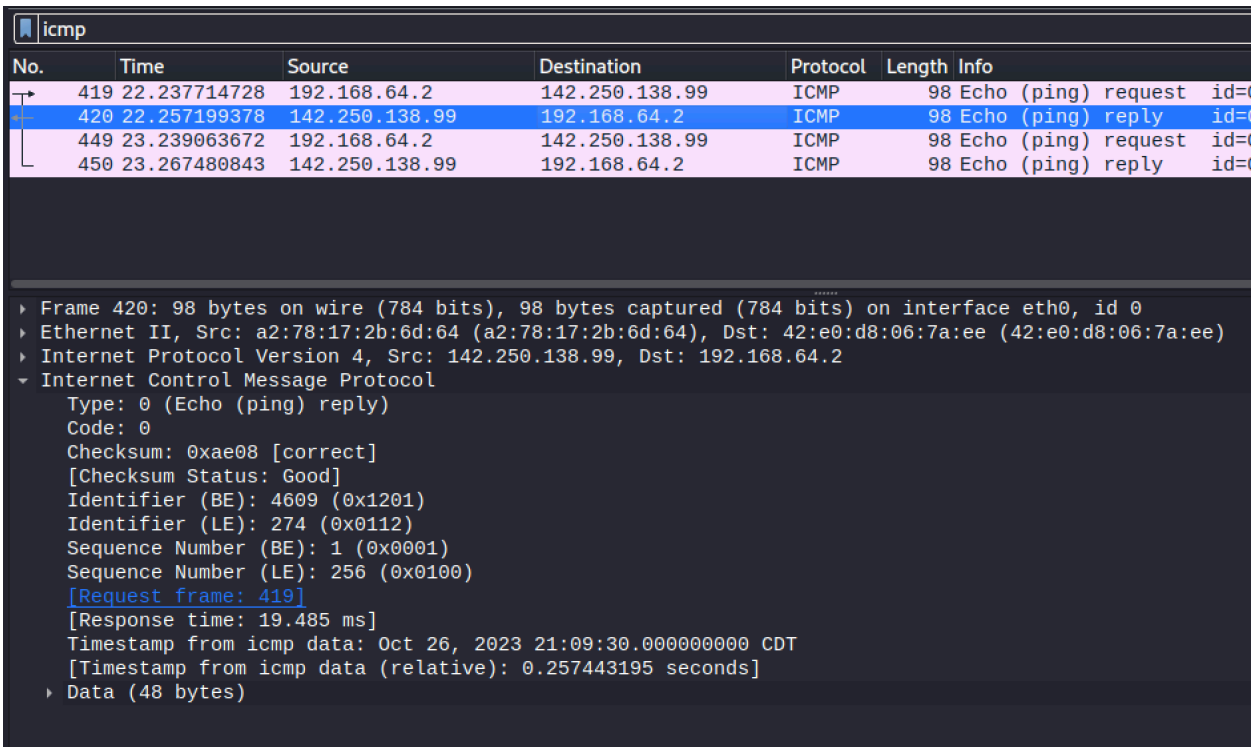
Next, after removing the DNS filter, I entered ICMP to filter for ICMP packets.



Here are the ICMP packets that were captured. There are multiple ICMP request and replies because I did not stop pinging immediately, therefore it repeated the ping.



Looking into the first ICMP request, the ICMP type is 8, which according to IBM's website "ICMP type and code IDs" https://www.ibm.com/docs/en/qsip/7.4?topic=applications-icmp-type-code-ids, means that the ICMP packet is an echo packet.

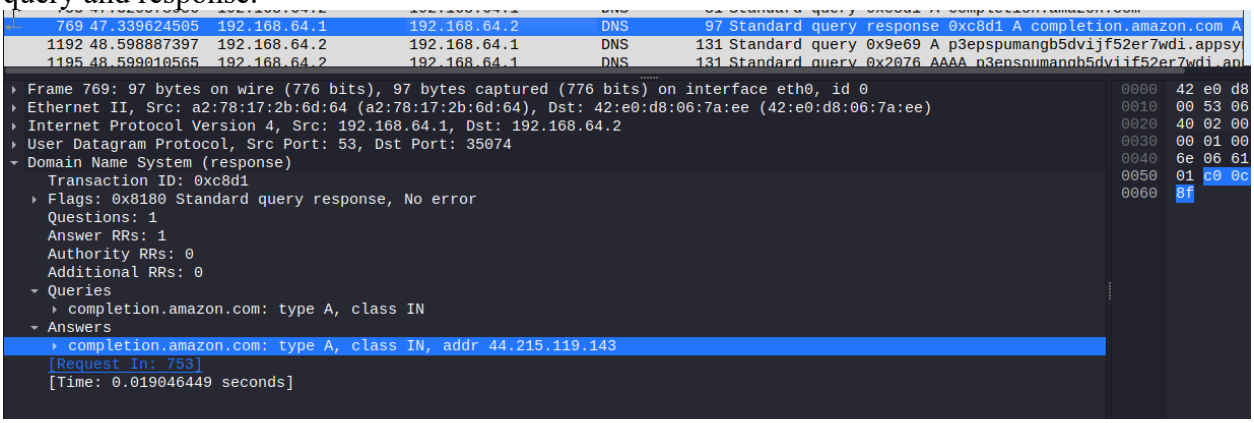


After the echo request packet, the following packet type was type 0, meaning echo reply. Because there was a reply, that proves that Google's servers are open and reachable.

Step *4: TLS*

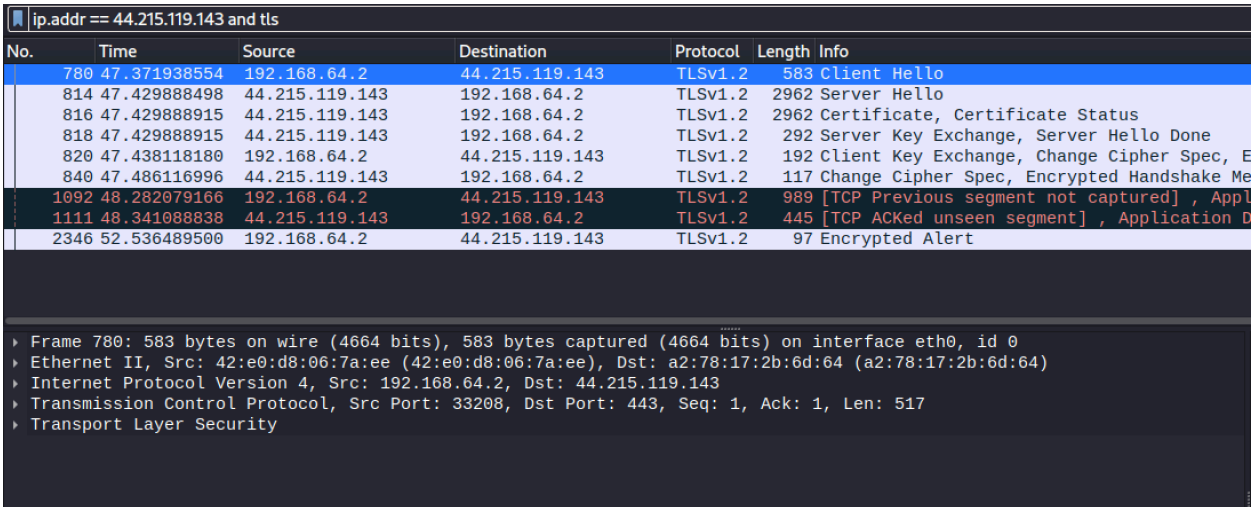
Next, I removed the ICMP filter and searched for the first TLS x.x. "Client Hello" that is associated with www.amazon.com. To find the TLS associated with amazon, I first needed to

find amazon's IP address. To do this I filtered for DNS packets again and found the Amazon query and response.

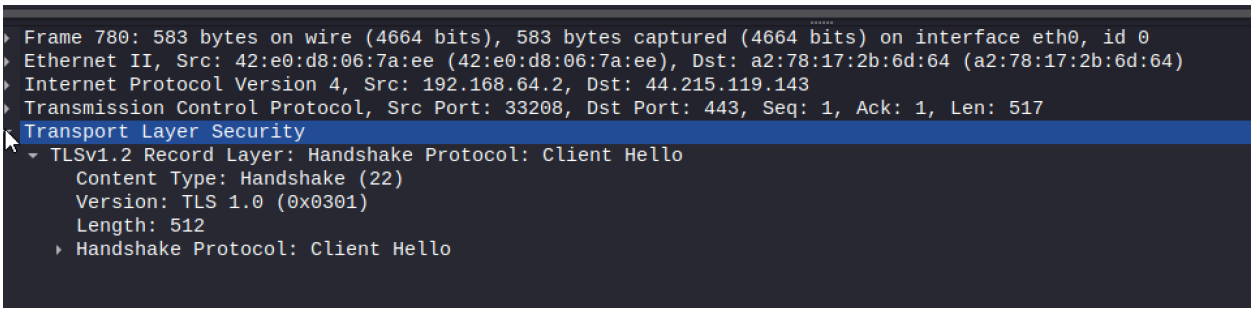


Once I found the DNS response associated with Amazon, I scrolled to the answers section and found Amazon's IP: 44.215.119.143.

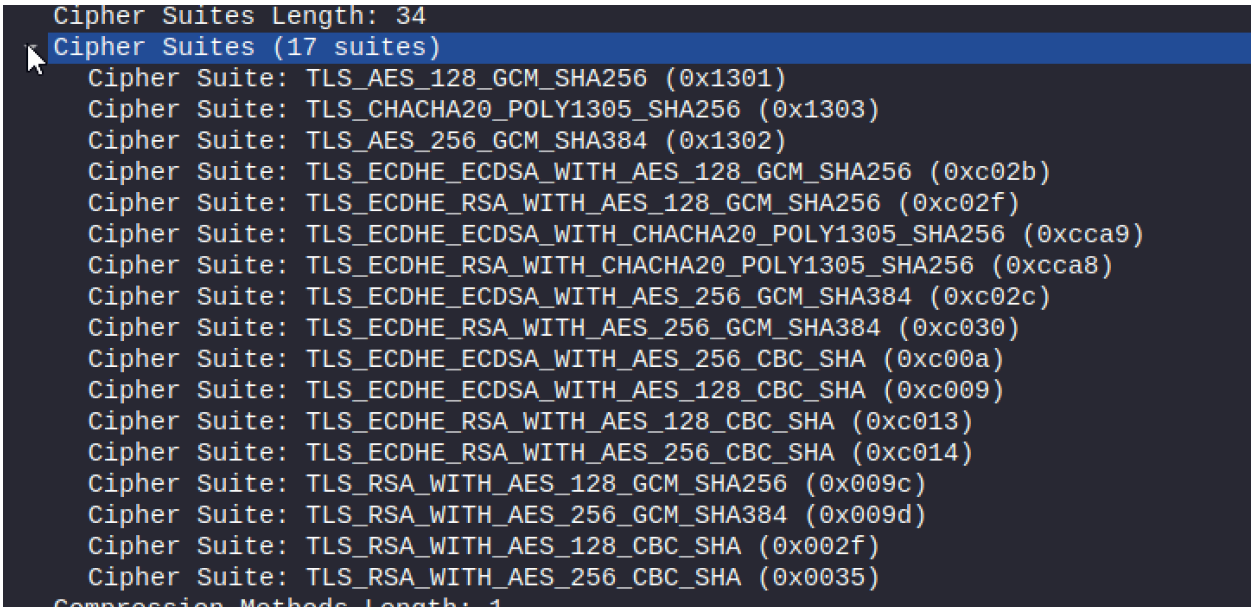
With this IP was able to specify my filter to specifically TLS packets associated with Amazon's IP. I used Wikipedia's website "Wireshark/Display filter -Wikiversity" https://en.wikiversity.org/wiki/Wireshark/Display_filter to learn how to filter by IP address on Wireshark.



To locate Amazon's 'Client Hello,' I used Wireshark's display filter. First, I filtered for Amazon's IP address, which was 44.215.119.143, to isolate packets associated with Amazon's server. Then, I specified 'tls' in the display filter to narrow down the results to TLS-encrypted packets. All together I entered "ip.addr == 44.215.119.143 and tls." Doing this, I was able to successfully filter for TLS packets associated with Amazon's IP address, and I was able to find the initial "Client Hello."



Next, on the "Client hello" packet, I expanded the "Transport Layer Security" segment.



And I then expanded the "Cipher Suites" segment to view the list of cipher suites supported.

Step *5: Hex Cheat Sheet*
In this step, I reviewed.

Hex File Header and ASCII Equivalent

File headers are used to identify a file by examining the first 4 or 5 bytes of its hexadecimal content.

Filetype	Start	Start ASCII Translation
ani	52 49 46 46	RIFF
au	2E 73 6E 64	snd
bmp	42 4D F8 A9	BM
bmp	42 4D 62 25	BMp%
bmp	42 4D 76 03	BMv
cab	4D 53 43 46	MSCF
dll	4D 5A 90 00	MZ
Excel	D0 CF 11 E0	
exe	4D 5A 50 00	MZP (inno)
exe	4D 5A 90 00	MZ
flv	46 4C 56 01	FLV
gif	47 49 46 38 39 61	GIF89a
gif	47 49 46 38 37 61	GIF87a
gz	1F 8B 08 08	
ico	00 00 01 00	
jpeg	FF D8 FF E1	
jpeg	FF D8 FF E0	JFIF
jpeg	FF D8 FF FE	JFIF
Linux bin	7F 45 4C 46	ELF
png	89 50 4E 47	PNG
msi	D0 CF 11 E0	
mp3	49 44 33 2E	ID3
mp3	49 44 33 03	ID3
OFT	4F 46 54 32	OFT2
PPT	D0 CF 11 E0	
PDF	25 50 44 46	%PDF
rar	52 61 72 21	Rar!
sfw	43 57 53 06/08	cws
tar	1F 8B 08 00	
tgz	1F 9D 90 70	
Word	D0 CF 11 E0	
wmv	30 26 B2 75	
zip	50 4B 03 04	PK

According to the hex cheat sheet, the PDF header values are 25, 50, 44, and 46.

Step *6: Lab 3 Packet Capture*

In this step, I needed to download the “Lab3 Capture” file and open it.

Lab 3

Due: Mon Oct 30, 2023 11:59pm

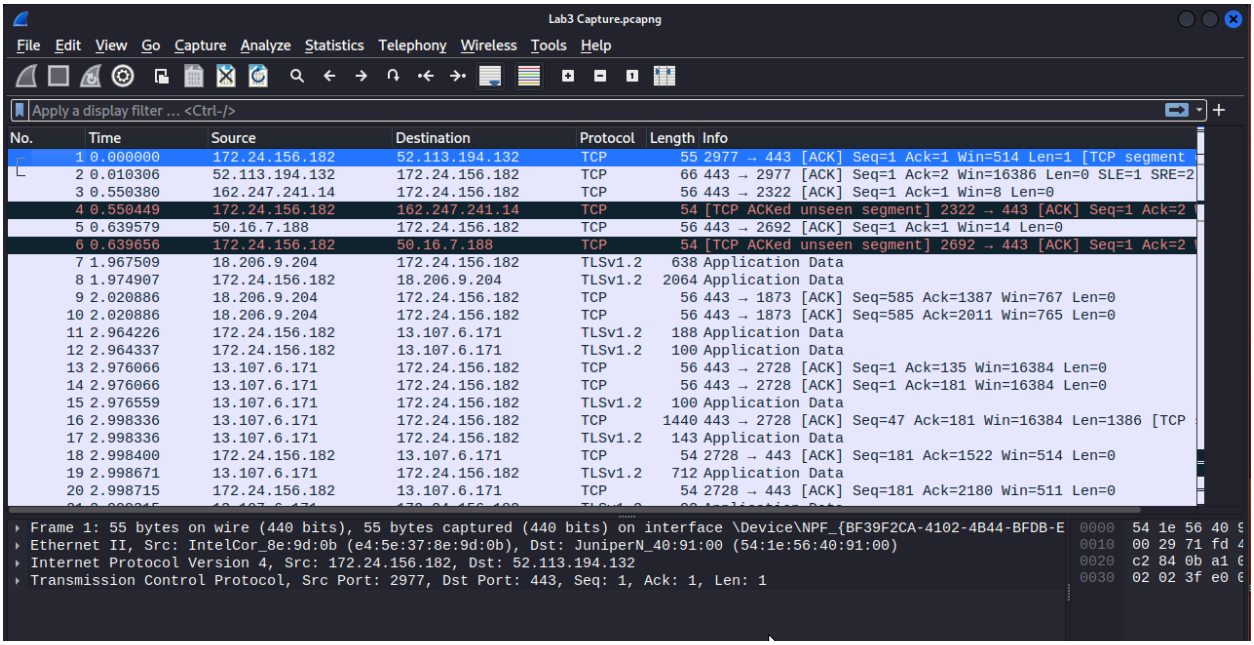
Attempt 1

In Progress
NEXT UP: Submit Assignment

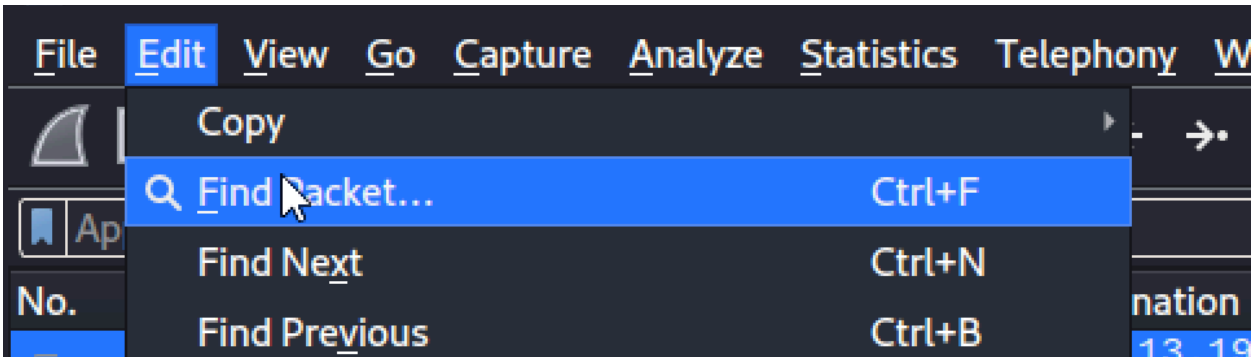
Lab3 Capture.pcapng

hex_file_and_regex_cheat_sheet.pdf

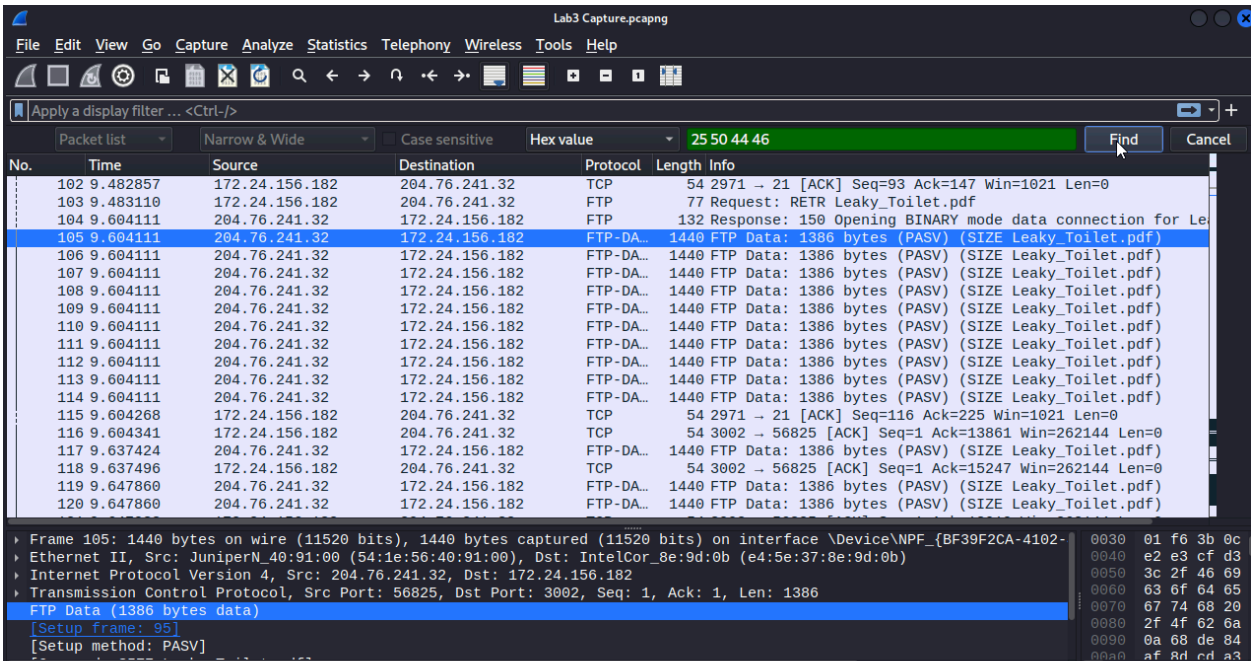
To begin, I opened Canvas and found the assignment. Once I opened the Lab 3 assignment, I was able to download the "Lab3 Capture" file.



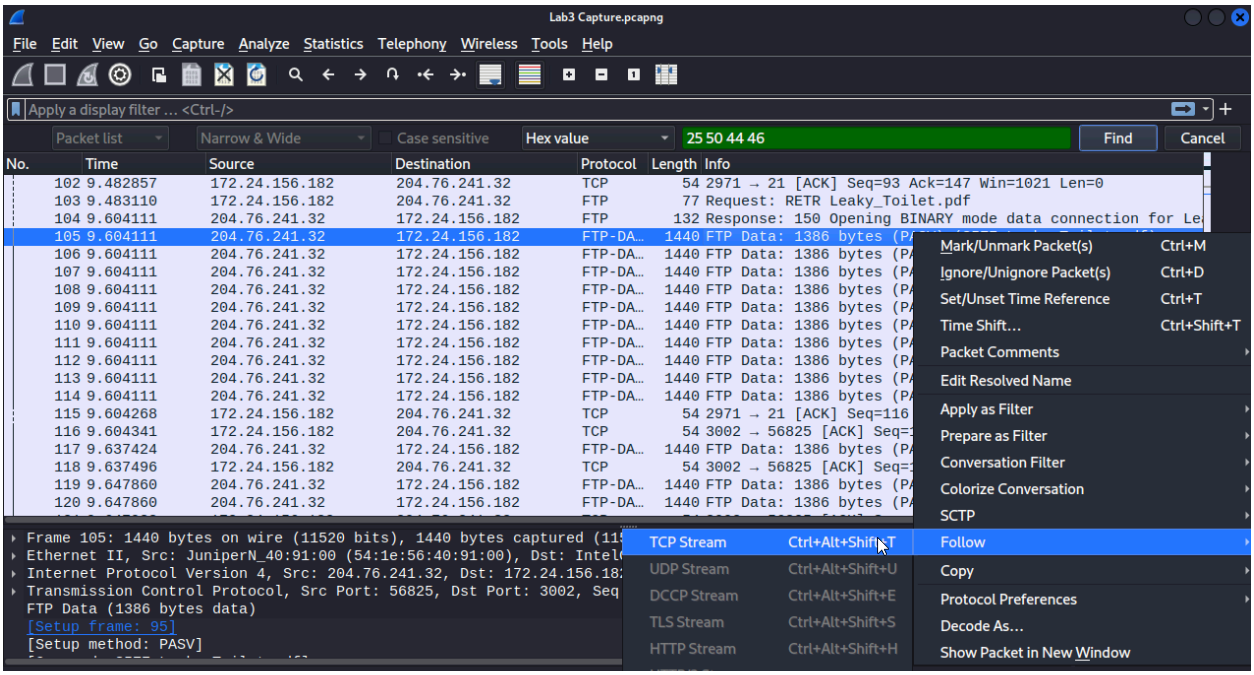
Next, I opened the packet capture file.



Selected Edit, then "Find Packet..."

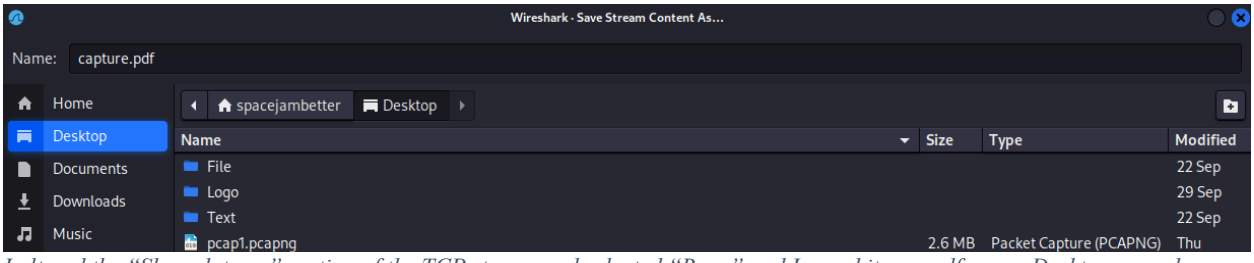


And I changed the display filter to "Hex value" and entered the header values from the cheat sheet (25, 50, 44, 46) and clicked 'Find."



I then selected the found packet and followed its TCP stream.

Screenshot of the TCP stream.



I altered the "Show data as" section of the TCP stream and selected "Raw," and I saved it as a pdf on my Desktop named "capture.pdf."

After saving the capture as a pdf, I opened my terminal to open the file.

Screenshot of me using the terminal to open the capture.pdf file.

Screenshot of the contents of the capture.pdf file. It appears to be an infographic on fixing leaky toilets. Very helpful!

Questions for the Lab

1. What is the Domain Name System (DNS)? What would happen if the DNS didn't function properly?

DNS is a protocol designed to convert human-friendly names to IP addresses. This facilitates the access of websites because, without DNS, users would have to memorize a site's IP address to access the website. If DNS were to not function properly, accessing website would be difficult as most people do not memorize their favorite website's IP address. Another potential issue with DNS is that if it is mapped incorrectly, users could be redirected to the wrong address. Regardless of the issue, DNS problems make it difficult to access a desired site.

2. What is the purpose of ICMP?

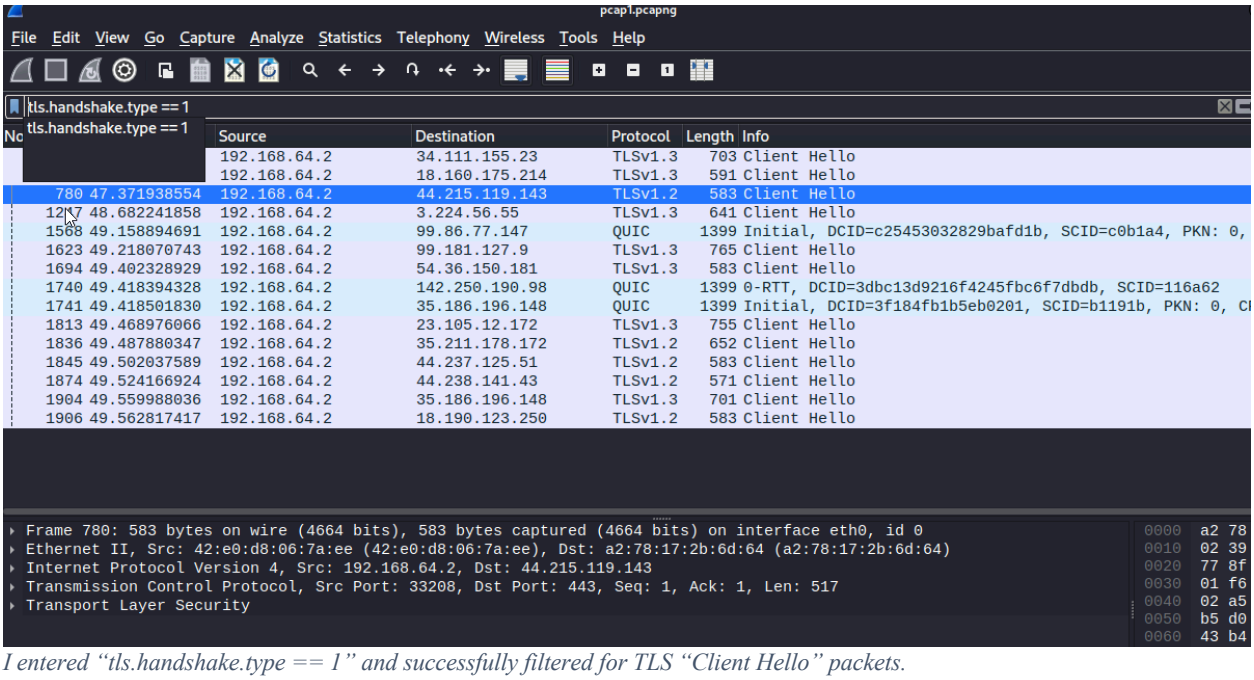
ICMP is used for network diagnostics and analyzing the network status. By using ICMP echo requests (pinging), one could view the availability of a specific port or server. In this lab, I pinged the google server to verify if it was available and reachable.

3. What is TLS? Which version did you observe in your packet capture?

TLS is a cryptographic protocol designed to secure and encrypt communication over a network. In this lab, I observed TLSv1.2. According to Gigamon Blog's article, "What is TLS 1.2 and Why Should You (Still) Care?" https://blog.gigamon.com/2021/07/14/what-is-tls-1-2-and-why-should-you-still-care/, TLSv1.2 provides increased security by using a single hash and both symmetric and asymmetric cryptography.

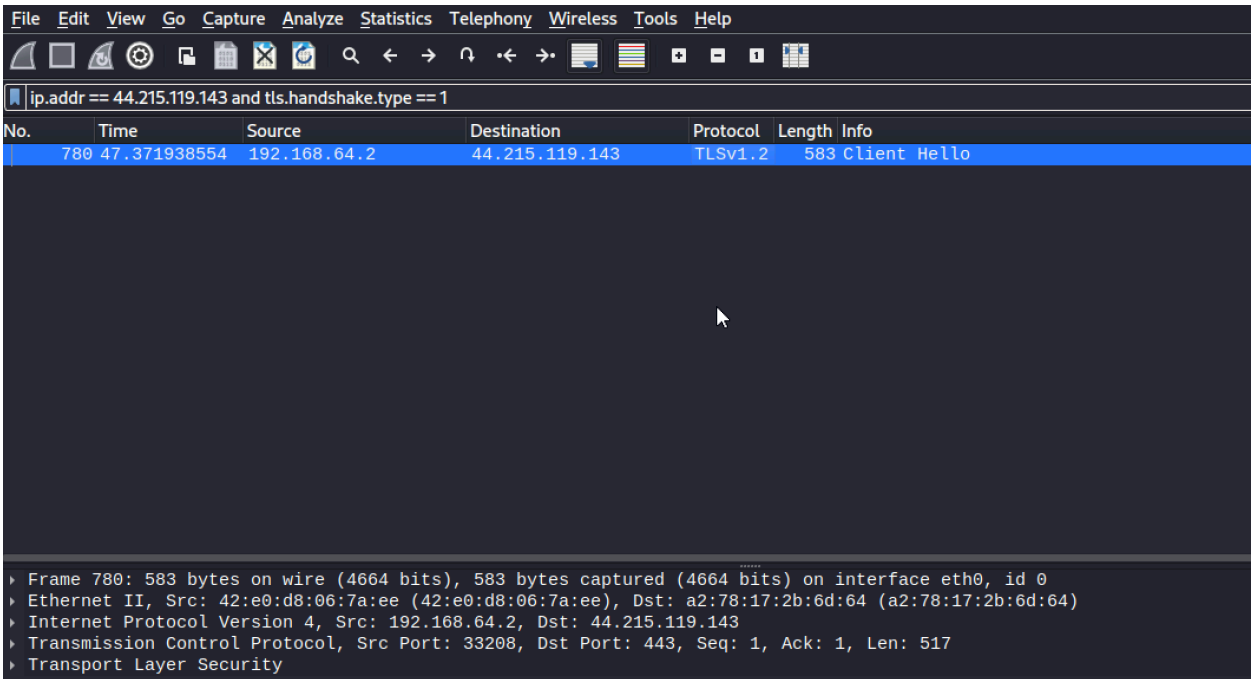
4. Is there a Wireshark filter that could have saved you time in finding the "Client Hello."

I had no problem finding the 'Client Hello" for Amazon because I used the "ip.addr == 44.215.119.143 and tls" filter to find TLS packets associated with Amazon's IP. But there is a better way to search for "Client Hello" packets. I read InsidePacket's webpage "Wireshark Filter for SSL Traffic" https://davidwzhang.com/2018/03/16/wireshark-filter-for-ssl-traffic/ and learned that I could filter for SSL "Client Hello" packets using the following filter "ssl.handshake.type == 1." Because SSL is similar to TLS (they are both cryptographic protocols that use a handshake), I decided to see if the filter works the same way if I replace SSL with TLS.



I entered "tls.handshake.type == 1" and successfully filtered for TLS "Client Hello" packets.

This is an effective way to filter for "Client Hello" packets for the entire network traffic, but if you're looking specific "Client Hello" packets to an IP address, there is another way.



I entered both the IP filter and the TLS filter to find the Client Hello associated with Amazon's IP.

5. What is FTP? How are you able to capture the contents of a .pdf using Wireshark in this example?

FTP (file transfer protocol) is a protocol designed to transfer files across the internet. Using Wireshark, I was able to capture an FTP packet labeled FTP-Data. Because FTP (File Transfer Protocol) is designed to transfer files across the internet, I knew that there was some sort of file transfer occurring. More specifically, I identified the type of file being transferred as a PDF because I filtered for the Hex Header values associated with PDFs (25 50 44 46). Next, I analyzed the file by following the TCP stream and converting the data to raw. When data is converted to "raw," it is converted into hexadecimal values which could be interpreted by a computer. Once the data was converted to raw, I was able to export the file as a pdf and open the image.

LIMITATIONS/CONCLUSION

In this lab I learned about Network sniffing through Wireshark. I learned about several protocols like FTP, ICMP, DNS, and TLS, and I learned about useful Wireshark filters which could facilitate traffic analysis. One limitation I had during the lab was I was unable to recreate the "no such name" DNS response on my Kali Linux virtual computer. It is likely that the Wireshark is configured differently on the machine as I was able to recreate the "no such name" through Wireshark on my Mac.

References

BlueCat Networks: “The top four DNS response codes and what they mean”
https://bluecatnetworks.com/blog/the-top-four-dns-response-codes-and-what-they-mean/
I used this website to understand the different DNS responses and what they mean.

IBM: “ICMP type and code IDs”
https://www.ibm.com/docs/en/qsip/7.4?topic=applications-icmp-type-code-ids
I used this website to understand the different type of ICMP types and what they mean.

Paul Browning: “Use Wireshark to Troubleshoot DNS Problems”
https://www.youtube.com/watch?v=vJsOteThRYA
This video gave me a reference to how my machine should react to a fake URL.

Gigamon Blog: “What is TLS 1.2 and Why Should You (Still) Care?”
https://blog.gigamon.com/2021/07/14/what-is-tls-1-2-and-why-should-you-still-care/
I used this article to understand TLSv1.2 and how it works.

Wikipedia: “Wireshark/Display filter -Wikiversity”
https://en.wikiversity.org/wiki/Wireshark/Display_filter
I used this website to learn how to filter by IP address on Wireshark.