Joseph Pennington (2912079)

EECS 565 Mini Project 2

Introduction

Wireshark is an extremely popular network protocol analyzer that allows one to see what is occurring on a network on an extremely detailed scale. This software is used across many industries such as non-profit organizations, government agencies, and educational institutions. Wireshark is primarily used as an open-source packet analyzer that intercepts and logs traffic that passes through a computer network. As the data is sent through the network, Wireshark captures and decodes each packet, allowing the user to see the data being sent.

For this project, we were asked to analyze the results of two different tasks. Using my apartment's Wi-Fi, the first task asked to analyze the results of filtering out our individual IP address on Wireshark to see sniffed packets from visiting Google.com. The second task was to exclude our MAC address and sniff for an extended period of time. From the sniffed packets, we were then asked to describe the various protocols and to attempt to identify the make or model of different devices.

Task 1

Task 1 had us only include our IP address to see what packets our computer sent and received when visiting Google.com. When originally navigating to Google.com while running Wireshark, I did not see any HTTP or HTTPS packets. Instead, I received many QUIC packets. This protocol is a form of TCP that is specifically used on the Google Chrome web browser that adds an additional layer of protection to one's data. However, when running the same experiment using Internet Explorer, I was able to sniff HTTP packets. Below is a screenshot of the HTTP packets. I originally saw TLSv1.2 and TCP traffic before the HTTP packets, followed by DNS traffic.

No.	Time	Source	Destination	Protocol	Length	Info
5894	5.171971	216.58.192.142	192.168.2.232	HTTP	582	HTTP/1.1 301 Moved Permanently (text/html)
5766	5.123930	192.168.2.232	216.58.192.142	HTTP	337	GET / HTTP/1.1
5491	4.452920	192.168.2.232	72.21.91.29	HTTP	288	GET /MFEwTzBNMEswSTAJBgUrDgMCGgUABBQQX6Z6gAidtSefNc6DC0OInqPHDQQUD4BhHIIxYdUvKOeNRji0LOHG2eICEArrhMcnws53%2F9qAdhtdaCw%3D HTTP/1.1
5405	3.884909	192.168.2.232	151.139.128.14	HTTP	285	GET /MFEwTzBNMEswSTAJBgUrDgMCGgUABBRDC9IOTxN6GmyRjyTl2n4yTUczyAQUjYxexFStiuF36Zv5mwXhuAGNYeECEHNFqRIcn00cf320aG0hohk%3D HTTP/1.1

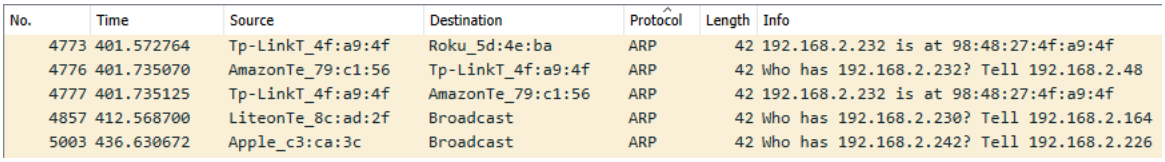
Task 2

Task 2 consisted of excluding our MAC address and letting Wireshark sniff the network for an extended period. I sniffed the network for thirty minutes. During these thirty minutes, the only devices that were actively being used were my computer that was running Wireshark and two iPhones. However, the results of the sniffing showed that many Wi-Fi enabled devices were

sending information. The most active devices were streaming devices, my computer, and my roommate's iPad Air. Below, I will briefly describe the most common protocols that were recorded and include a screenshot of the output.

ARP (Address Resolution Protocol)

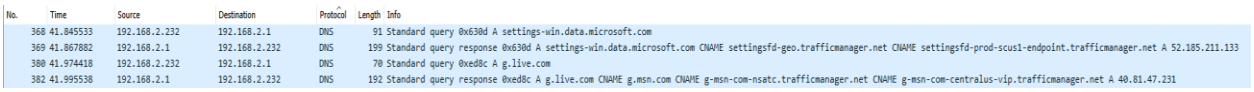
ARP is used to dynamically discover the mapping between the protocol and the hardware address. This usually maps the IP address to an Ethernet address. These are typically seen at the beginning of a conversation. The results showed that mostly my Apple TV and Roku were responsible for this protocol.



No.	Time	Source	Destination	Protocol	Length	Info
4773	401.572764	Tp-LinkT_4f:a9:4f	Roku_5d:4e:ba	ARP	42	192.168.2.232 is at 98:48:27:4f:a9:4f
4776	401.735070	AmazonTe_79:c1:56	Tp-LinkT_4f:a9:4f	ARP	42	Who has 192.168.2.232? Tell 192.168.2.48
4777	401.735125	Tp-LinkT_4f:a9:4f	AmazonTe_79:c1:56	ARP	42	192.168.2.232 is at 98:48:27:4f:a9:4f
4857	412.568700	LiteonTe_8c:ad:2f	Broadcast	ARP	42	Who has 192.168.2.230? Tell 192.168.2.164
5003	436.630672	Apple_c3:ca:3c	Broadcast	ARP	42	Who has 192.168.2.242? Tell 192.168.2.226

DNS (Domain Name System)

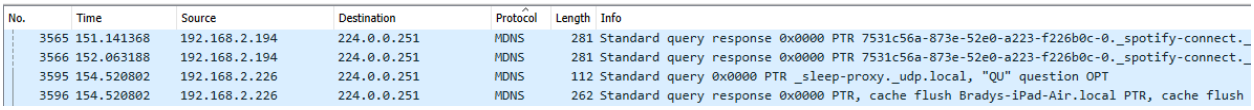
DNS translates domain names into IP addresses which allow internet browsers to load internet resources. The most common device with this protocol was my computer. It was sending messages to various Google and Microsoft destinations.



No.	Time	Source	Destination	Protocol	Length	Info
368	41.845533	192.168.2.232	192.168.2.1	DNS	91	Standard query 0x630d A settings-win.data.microsoft.com
369	41.867882	192.168.2.1	192.168.2.232	DNS	199	Standard query response 0x630d A settings-win.data.microsoft.com CNAME settingsfd-geo.trafficmanager.net CNAME settingsfd-prod-scus1-endpoint.trafficmanager.net A 52.185.211.133
380	41.974418	192.168.2.232	192.168.2.1	DNS	70	Standard query 0xed8c A g.live.com
382	41.995538	192.168.2.1	192.168.2.232	DNS	192	Standard query response 0xed8c A g.live.com CNAME g.msn.com CNAME g-msn-com-nsatc.trafficmanager.net CNAME g-msn-com-centralus-vip.trafficmanager.net A 40.81.47.231

MDNS (Multicast Domain Name System)

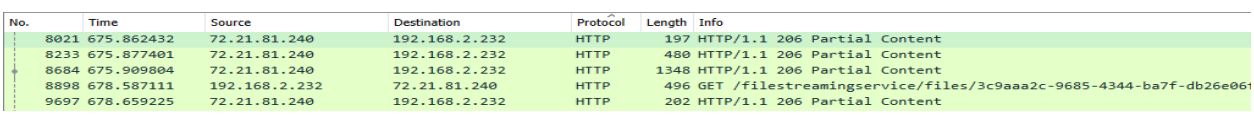
MDNS resolves hostnames to IP addresses in smaller networks. The most common devices with this protocol were my streaming devices and my roommate's iPad Air.



No.	Time	Source	Destination	Protocol	Length	Info
3565	151.141368	192.168.2.194	224.0.0.251	MDNS	281	Standard query response 0x0000 PTR 7531c56a-873e-52e0-a223-f226b0c-0._spotify-connect._
3566	152.063188	192.168.2.194	224.0.0.251	MDNS	281	Standard query response 0x0000 PTR 7531c56a-873e-52e0-a223-f226b0c-0._spotify-connect._
3595	154.520802	192.168.2.226	224.0.0.251	MDNS	112	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QU" question OPT
3596	154.520802	192.168.2.226	224.0.0.251	MDNS	262	Standard query response 0x0000 PTR, cache flush Bradys-iPad-Air.local PTR, cache flush

HTTP (Hyper Text Transfer Protocol)

HTTP is a text-based request-response protocol. A client performs a request and a server returns a response. My computer made the majority of these requests.



No.	Time	Source	Destination	Protocol	Length	Info
8021	675.862432	72.21.81.240	192.168.2.232	HTTP	197	HTTP/1.1 206 Partial Content
8233	675.877401	72.21.81.240	192.168.2.232	HTTP	480	HTTP/1.1 206 Partial Content
8684	675.909804	72.21.81.240	192.168.2.232	HTTP	1348	HTTP/1.1 206 Partial Content
8898	678.587111	192.168.2.232	72.21.81.240	HTTP	496	GET /filestreamingservice/files/3c9aaa2c-9685-4344-ba7f-db26e06
9697	678.659225	72.21.81.240	192.168.2.232	HTTP	202	HTTP/1.1 206 Partial Content

TCP (Transmission Control Protocol)

TCP was the most common protocol recorded. TCP provides stream-based connection-oriented transfer of data. It allows for programs and devices to exchange messages over a network. It specifically sends packets and ensures that they are delivered correctly. The majority of the output originated from my computer. Either my computer was sending information to websites or receiving information.

No.	Time	Source	Destination	Protocol	Length	Info
3373	103.262260	192.168.2.232	129.237.11.132	TCP	55	[TCP Keep-Alive] 60744 → 443 [ACK] Seq=1 Ack=1 Win=513 Len=1
3374	103.293291	129.237.11.132	192.168.2.232	TCP	54	[TCP Keep-Alive ACK] 443 → 60744 [ACK] Seq=1 Ack=2 Win=2560 Len=0
3377	103.440647	192.168.2.232	54.210.196.128	TCP	55	[TCP Keep-Alive] 60750 → 443 [ACK] Seq=1 Ack=1 Win=513 Len=1

QUIC

QUIC is a form of TCP that provides an interface for sending data between two destinations. QUIC is primarily used on the Google Chrome web browser. These protocols originated from my computer mostly likely because I was using the Chrome browser.

No.	Time	Source	Destination	Protocol	Length	Info
61711	923.506171	192.168.2.232	172.217.4.78	QUIC	1392	Handshake, DCID=842a9b81c594ef53
61712	923.507447	192.168.2.232	172.217.4.78	QUIC	115	Handshake, DCID=842a9b81c594ef53
61713	923.507633	192.168.2.232	172.217.4.78	QUIC	112	Protected Payload (KP0), DCID=842a9b81c594ef53

UDP (User Datagram Protocol)

UDP allows computer applications to send messages as datagrams to other hosts on an IP network. However, it does not guarantee that the message will be delivered. This protocol is usually used by time-sensitive applications. Most of these protocols came from the router itself.

No.	Time	Source	Destination	Protocol	Length	Info
3105	55.151022	fe80::19d6:ee55:cb4…	ff02::c	UDP	718	59073 → 3702 Len=656
3109	55.952898	192.168.2.232	239.255.255.250	UDP	698	59072 → 3702 Len=656
3177	61.234755	192.168.2.1	233.89.188.1	UDP	46	50241 → 10001 Len=4
3178	61.234755	192.168.2.1	255.255.255.255	UDP	46	50241 → 10001 Len=4
3223	71.781864	192.168.2.1	233.89.188.1	UDP	46	50241 → 10001 Len=4

Conclusion

Wireshark is an open-source packet sniffing software that is used across many industries. It allows users to analyze the packets being sent on a network. This project had us analyze the results of two different experiments. The biggest takeaway from the experiments is that most of my devices were sending lots of data even though I was not actively using them. This demonstrates that it is important to understand the basics of network security in order to better protect oneself and one’s data.

References

Wireshark Website, https://wiki.wireshark.org/FrontPage