

Introduction - Alexis Xavier 33791124

Warning: This experiment may contain personal details on my own home network

Within this topic, I wanted to conduct my own personal experiments in order to gain a greater understanding of Networking as well as expand on my understanding of cybersecurity overall. This module presented me with an opportunity to investigate further into this. Mainly used two primary tools: nmap and Wireshark.

Nmap - Nmap is commonly known as a network scanner and it is short for network mapper. It is essentially used to discover hosts and services on a computer network through sending packets and analyzing the responses. It is a tool that can also scan IP addresses and ports in a network as well to help detect if certain applications have been installed.

Wireshark - Wireshark is most commonly known to be a network packet analyzer. It shows the user captured packet data, and tries to show as much detail as possible. It helps understand more about what is going on inside a network

Summary of Experiment

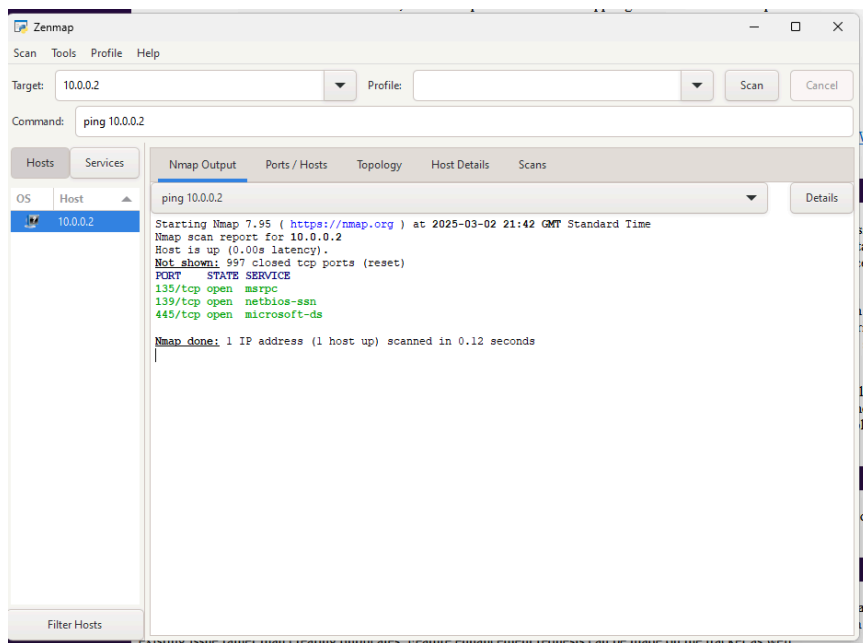
During this experiment, my goal was to be able to learn more about my own personal home network as well as to help expand my understanding of these tools seeing how they are crucial for later use in the cybersecurity field. They are essential for carrying out penetration testing and therefore I felt that it was important to understand more about.

Essentially, I installed both wireshark and nmap on my own home PC too and analyzed what was going on. I searched up some tutorials on YouTube in order to help get me started in understanding more about the software. At some point, I also wanted to expand this using a more popular network that was not my own, however I felt that it may have been a breach of legislation and therefore I chose not to so the findings of hosts on my networks are limited.

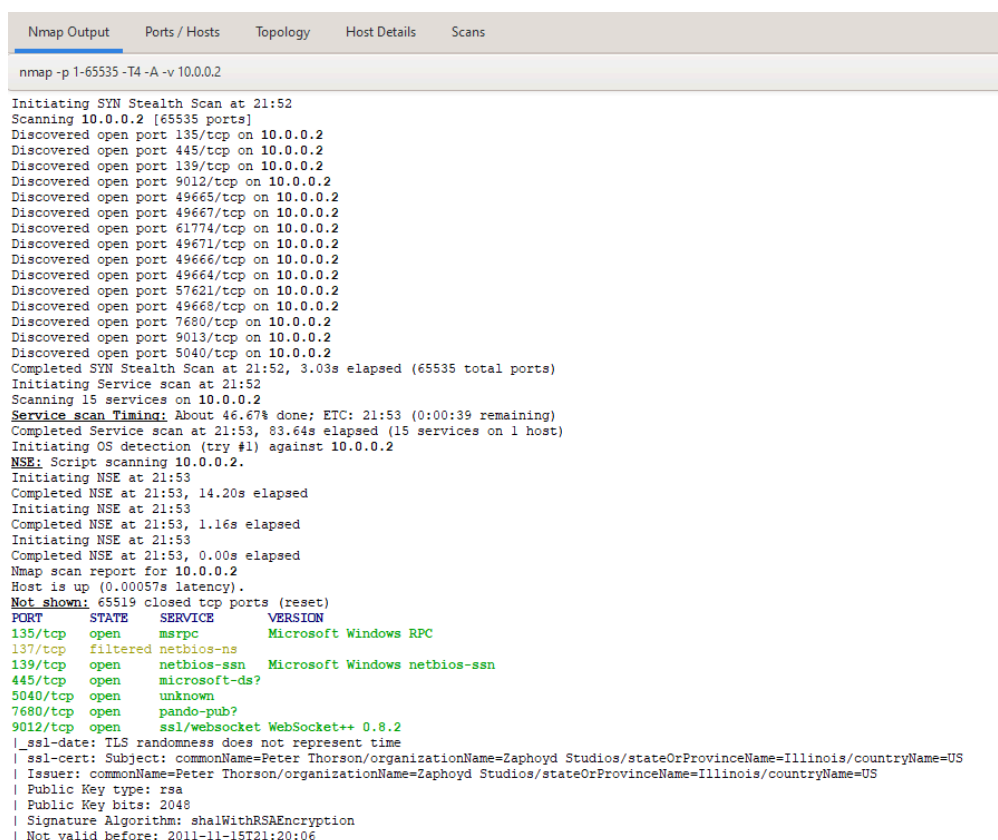
Alongside with this, I also wanted to explore my own router and found out there to be some really interesting functionality in which my router provided.

Phase 1

My first intention was to be able to successfully install nmap and use its application. I also have nmap installed on Kali Linux however since it was contained within a Virtual Machine, I felt like I wouldnt get the readings that I intended to gain.



When I first opened Nmap, I was presented with an application called ZenMap, at first i was confused, however I later came to know that this is a more visual application with Nmap to help reduce the complexity that a lot of new users may encounter.



The command: `nmap -p 1-65535 -T4 -A -v 10.0.0.2`

This was from the tutorials in which i watched and this command essentially scans all 65,535 TCP ports i.e Full port Scan.

-T4 means that it was level 4 aggressive, and that it would try to be a faster scan, however it also meant that this scan was detectable.

-A meant that it enabled aggressive scan options such as OS detection, version detection, script scanning and traceroute.

-v = Provides a more detailed output in real time.

10.0.0.2 = Target IP address.

Findings showed that certain ports were open and accessible.

Port	Service	Description
135/tcp	msprc	Microsoft RPC service — typical on Windows machines.
139	netbios-ssn	Used for Windows file and printer sharing (NetBIOS Session Service).
445	microsoft-ds	SMB file sharing over TCP. Often used in Windows networks.
9012	pando-pub?	Custom/ Unknown service
9013	websocket	Running websocket +++ version 0.8.2. This tells us that 9013 is using SSL/TLS encryption. Self-signed certificate

For 137, there is a firewall blocking this port since nmap did not receive a response.

```

Public Key type: rsa
Public Key bits: 2048
Signature Algorithm: sha1WithRSAEncryption
Not valid before: 2011-11-15T21:20:06
Not valid after: 2012-11-14T21:20:06
MD5: fe49:23f5:2b50:6c58:5d19:0c0d:0176:daa7
SHA-1: 8c44:2ddf:d594:5b22:204c:17d0:375a:a8ef:6956:ce8d
013/tcp open  websocket      WebSocket++ 0.8.2
9664/tcp open  msrpc          Microsoft Windows RPC
9665/tcp open  msrpc          Microsoft Windows RPC
9666/tcp open  msrpc          Microsoft Windows RPC
9667/tcp open  msrpc          Microsoft Windows RPC
9668/tcp open  msrpc          Microsoft Windows RPC
9671/tcp open  msrpc          Microsoft Windows RPC
7621/tcp open  unknown
1774/tcp open  unknown
fingerprint-strings:
  NULL:
  ("type":"Tier1","version":"1.0")
- service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
F-Port61774-TCP:V=7.95%I=7%D=3/2%Time=67C4D308%P=i686-pc-windows-windows%
E:r(NULL,22,{"type\":"Tier1","version\":"1.0"}\r\n");
service type: general purpose
running: Microsoft Windows 10|11
S_CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows_11
S_details: Microsoft Windows 10 1607 - 11 23H2
ptime_guess: 0.327 days (since Sun Mar 2 14:03:18 2025)
etwork Distance: 0 hops
CP Sequence Prediction: Difficulty=249 (Good luck!)
P ID Sequence Generation: Incremental|
ervice Info: OS: Windows; CPE: cpe:/o:microsoft:windows

ost script results:
  smb2-time:
    date: 2025-03-02T21:53:34
  - start_date: N/A
  - smb2-security-mode:
    3.1:1:
  - Message signing enabled but not required

SE: Script Post-scanning.
nitiating NSE at 21:53
ompleted NSE at 21:53, 0.00s elapsed
nitiating NSE at 21:53
ompleted NSE at 21:53, 0.00s elapsed
nitiating NSE at 21:53
ompleted NSE at 21:53, 0.00s elapsed

```

This provides a tail end of the Nmap scan with some deeper system level information.

I then conducted another scan later on within the day, as it was only me connected to the router at the time. This time the command line was: `nmap -F -n -Pn 10.0.0.1/24`

Target: 10.0.0.1/24 Profile:

Command: nmap -F -n -Pn 10.0.0.1/24

Hosts Services

Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

10.0.0.1

10.0.0.2

10.0.0.3

10.0.0.5

10.0.0.11

nmap -F -n -Pn 10.0.0.1/24

PORT STATE SERVICE

49152/tcp open unknown

MAC Address: 1E:7E:1F:F5:B4:92 (Unknown)

Nmap scan report for 10.0.0.11

Host is up (0.086s latency).

Not shown: 72 closed tcp ports (reset)

PORT STATE SERVICE

9/tcp filtered discard

13/tcp filtered daytime

37/tcp filtered time

81/tcp filtered hosts2-ns

88/tcp open kerberos-sec

427/tcp filtered svrloc

513/tcp filtered login

514/tcp filtered shell

544/tcp filtered kshell

548/tcp filtered afp

873/tcp filtered rsync

1026/tcp filtered LSA-or-nterm

1028/tcp filtered unknown

1110/tcp filtered nfsd-status

1433/tcp filtered ms-sql-s

2049/tcp filtered nfs

2121/tcp filtered cproxy-ftp

2717/tcp filtered pn-requester

3000/tcp filtered ppp

5101/tcp filtered admdog

5900/tcp open vnc

7070/tcp open realserver

8000/tcp filtered http-alt

8008/tcp filtered http

8009/tcp filtered ajpl3

8443/tcp filtered https-alt

9100/tcp filtered jetdirect

9999/tcp filtered abyss

MAC Address: EC:35:86:30:C5:F6 (Apple)

Nmap scan report for 10.0.0.2

Host is up (0.000031s latency).

Not shown: 97 closed tcp ports (reset)

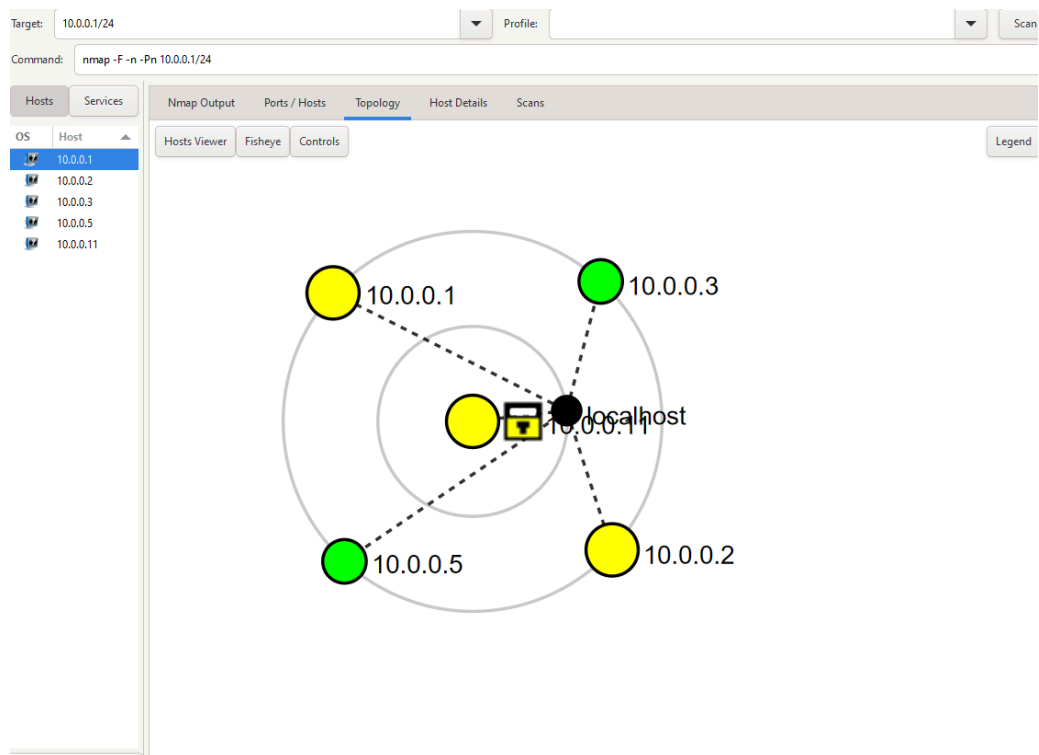
PORT STATE SERVICE

135/tcp open msrpc

139/tcp open netbios-ssn

445/tcp open microsoft-ds

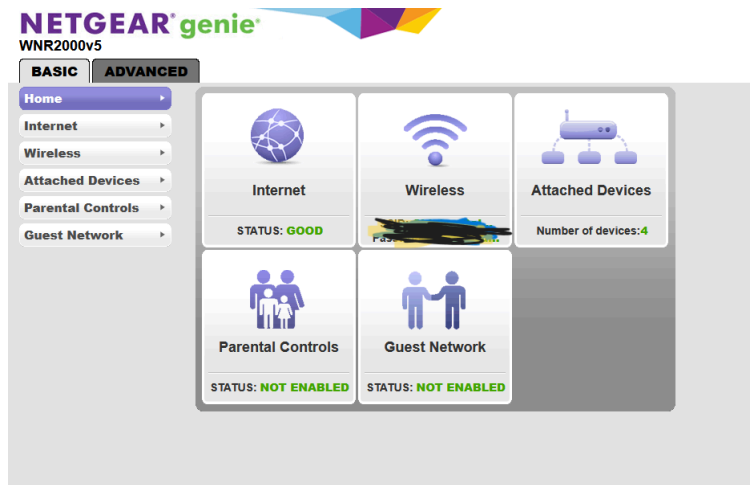
Nmap done: 256 IP addresses (5 hosts up) scanned in 8.99 seconds



Findings:

These findings showed me that there were around 5 live hosts on the subnet. The green means that there is good visibility however the yellow meant that they were scanned with limited information or mostly filtered ports.

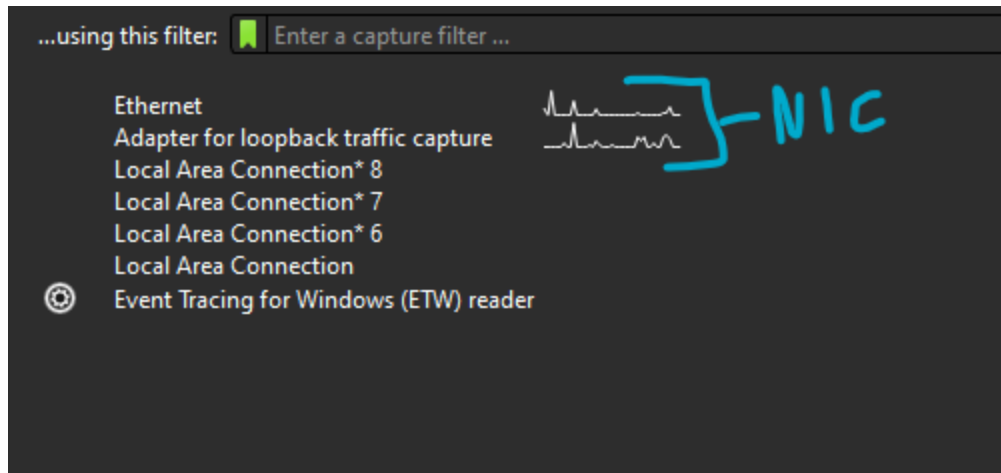
After this experiment, I went and checked on my own home router too see if the reading from nmap was accurate:



As a result of me wanting to dive deeper, I found a function of my router that would essentially route packets towards my own PC, this meant that when I would run wireshark, I wouldn't be getting only the PC packets, but also the packets for the other hosts. This was known as port forwarding. Although I wanted to test it out, I felt like if I start tinkering my family wouldn't be happy with the wifi being reset if something went wrong.

Phase 2

Within this phase, I officially installed wireshark and had a look around the packets in which were being sent and received. When using wireshark, similarly to nmap, I watched a youtube tutorial in order to help get me started.



First I opened up Wireshark and it showed me what I was connected to, from there I decided to have a look at my ethernet.

No.	Time	Source	Destination	Protocol	Length	Info
17	3.226958	10.0.0.2	2.21.35.219	TCP	54	54610 → 443 [ACK] Seq=1 Ack=25 Win=1024 Len=0
18	3.226974	2.21.35.219	10.0.0.2	TCP	62	443 → 54630 [FIN, ACK] Seq=25 Ack=1 Win=0 Len=0
19	3.226983	10.0.0.2	2.21.35.219	TCP	54	54630 → 443 [ACK] Seq=1 Ack=26 Win=1024 Len=0
20	3.725434	13.107.6.254	10.0.0.2	TCP	62	443 → 54637 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	4.168644	40.90.136.180	10.0.0.2	TCP	62	443 → 54631 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	5.266092	20.200.159.71	10.0.0.2	TCP	62	443 → 54632 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	5.963327	10.0.0.2	224.0.0.251	MDNS	70	Standard query 0x0000 A wpad.local, "QI" question
24	5.963471	10.0.0.2	224.0.0.251	MDNS	70	Standard query 0x0000 AAAA wpad.local, "QI" question
25	5.963638	10.0.0.2	224.0.0.252	LLMNR	64	Standard query 0xad2 A wpad
26	5.963738	10.0.0.2	224.0.0.252	LLMNR	64	Standard query 0x364b AAAA wpad
27	6.027012	52.113.190.254	10.0.0.2	TCP	62	443 → 54635 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	6.040920	142.250.200.10	10.0.0.2	UDP	78	443 → 54954 Len=32
29	6.054180	10.0.0.2	142.250.200.10	UDP	75	54954 → 443 Len=33
30	6.142193	95.181.63.227	10.0.0.2	TLSv1.2	78	Application Data
31	6.142220	10.0.0.2	95.181.63.227	TCP	54	54622 → 443 [ACK] Seq=1 Ack=25 Win=1024 Len=0
32	6.142257	95.181.63.227	10.0.0.2	TCP	62	443 → 54622 [FIN, ACK] Seq=25 Ack=1 Win=2728 Len=0
33	6.142265	10.0.0.2	95.181.63.227	TCP	54	54622 → 443 [ACK] Seq=1 Ack=26 Win=1024 Len=0
34	6.161814	10.0.0.4	224.0.0.251	MDNS	84	Standard query response 0x0000 A, cache flush 10.0.0.4
35	6.380564	10.0.0.2	224.0.0.252	LLMNR	64	Standard query 0x364b AAAA wpad
36	6.380609	10.0.0.2	224.0.0.252	LLMNR	64	Standard query 0xad2 A wpad
37	6.380706	10.0.0.2	10.0.0.255	NDNS	92	Name query NB WPAD:00
38	6.974413	10.0.0.2	224.0.0.251	MDNS	70	Standard query 0x0000 A wpad.local, "QI" question
39	6.974541	10.0.0.2	224.0.0.251	MDNS	70	Standard query 0x0000 AAAA wpad.local, "QI" question
40	7.120695	10.0.0.4	224.0.0.251	MDNS	84	Standard query response 0x0000 A, cache flush 10.0.0.4
41	7.145838	10.0.0.2	10.0.0.255	NDNS	92	Name query NB WPAD:00
42	7.545667	10.0.0.1	224.0.0.1	IGMPv3	62	Membership Query, general
43	7.569282	10.0.0.2	224.0.0.22	IGMPv3	54	Membership Report / Join group 224.0.0.251 for any sources
44	7.897709	10.0.0.2	10.0.0.255	NDNS	92	Name query NB WPAD:00

Frame 1: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface 'DeviceNPF_{438CE94F-F169-41A0-9813-97D64185C083}', Channel 15, Ethernet II, Src: Intel(R) Ethernet Controller (P0-P3) (80:00:00:00:00:00), Dst: Intel(R) Ethernet Controller (P0-P3) (80:00:00:00:00:00)	0000 01 00 5e 00 00 0b de ea 2b 38 c0 3d 08 00 45 00	...N +B +E
Ethernet II, Src: Intel(R) Ethernet Controller (P0-P3) (80:00:00:00:00:00), Dst: Intel(R) Ethernet Controller (P0-P3) (80:00:00:00:00:00)	0000 00 5f 57 00 00 02 11 2e 38 00 00 00 00 00 00	...M
User Datagram Protocol, Src Port: 5553, Dst Port: 5553	0000 00 fb 14 e9 14 49 00 45 c6 91 00 0a 00 00 00 02	...E
Multicast Domain Name System (query)	0000 00 00 00 00 00 00 5f 67 6f 67 6c 65 63 61 61	...googleca
	0000 73 04 5f 74 43 78 09 6c 6f 63 61 6c 00 00 0c	...scop local
	0000 00 01 0f 32 33 33 36 33 37 44 45 04 5f 73 75	..._2336 370E _su
	0000 62 c8 0c 00 0c 00 01	...b.....

When I pressed ethernet, my computer began picking up a variety of packets. Once I felt like I had enough data, I then pressed the stop button. Each row that is currently being displayed on the screen is shown as a single packet.

So first I clicked on statistics and then clicked on conversations, this brought me to a page that shows me all of the PCAP that I have just captured.

Conversation Settings

☐ Name resolution

☐ Absolute start time

☐ Limit to display filter

Copy

Follow Stream...

Graph...

Protocol

☐ Bluetooth

☐ BPV7

☐ DCCP

☒ Ethernet

☐ FC

☐ FDDI

☐ IEEE 802.11

☐ IEEE 802.15.4

☒ IPv4

☒ IPv6

☐ IPX

☐ JXTA

☐ LTP

Ethernet · 15

IPv4 · 72

IPv6 · 1

TCP · 50

UDP · 52

Address A	Address B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
4eea2b38:c03d	01:00:5e:00:00:16	4	248 bytes	11	4	248 bytes	0	0 bytes	20.968175	141.0000	14 bits/s	0 bits/s
4eea2b38:c03d	01:00:5e:00:00:fb	7	709 bytes	0	7	709 bytes	0	0 bytes	0.000000	162.1943	34 bits/s	0 bits/s
4eea2b38:c03d	01:00:5e:7fff:fa	12	248 bytes	9	12	248 bytes	0	0 bytes	9.540900	30.2275	530 bits/s	0 bits/s
4eea2b38:c03d	33:33:00:00:00:16	2	180 bytes	14	2	180 bytes	0	0 bytes	160.883957	2.2091	651 bits/s	0 bits/s
74a7:ea:76:80:c4	01:00:5e:00:00:16	9	558 bytes	8	9	558 bytes	0	0 bytes	8.581493	206.0798	21 bits/s	0 bits/s
74a7:ea:76:80:c4	01:00:5e:00:00:fb	2	168 bytes	5	2	168 bytes	0	0 bytes	6.163814	0.9569	1404 bits/s	0 bits/s
74a7:ea:76:80:c4	f0:2f:74:15:42:75	57	8 kB	13	33	6 kB	24	2 kB	54.405228	165.7563	308 bits/s	98 bits/s
a0:63:91:78:0d:d2	01:00:5e:00:00:01	10	620 bytes	6	10	620 bytes	0	0 bytes	7.543667	224.5504	22 bits/s	0 bits/s
a0:63:91:78:0d:d2	01:00:5e:00:00:16	10	700 bytes	10	10	700 bytes	0	0 bytes	10.175364	223.8944	25 bits/s	0 bits/s
f0:2f:74:15:42:75	01:00:5e:00:00:16	30	2 kB	7	30	2 kB	0	0 bytes	7.569202	230.5041	56 bits/s	0 bits/s
f0:2f:74:15:42:75	01:00:5e:00:00:fb	6	454 bytes	3	6	454 bytes	0	0 bytes	5.963327	167.6308	21 bits/s	0 bits/s
f0:2f:74:15:42:75	01:00:5e:00:00:fc	4	256 bytes	4	4	256 bytes	0	0 bytes	5.963638	0.4230	4841 bits/s	0 bits/s
f0:2f:74:15:42:75	01:00:5e:7fff:fa	10	2 kB	12	10	2 kB	0	0 bytes	54.200163	163.9379	99 bits/s	0 bits/s
f0:2f:74:15:42:75	a0:63:91:78:0d:d2	33,984	27 MB	1	12,779	3 MB	21,205	24 MB	0.407227	236.2275	95 kbps	806 kbps
f0:2f:74:15:42:75	ff:ff:ff:ff:ff:ff	14	1 kB	2	14	1 kB	0	0 bytes	0.707784	210.0444	47 bits/s	0 bits/s

This allows me to see conversations, in particular, I look at TCP conversations and see any trends which may occur between my computer and another destination.

Ethernet · 15	IPv4 · 72	IPv6 · 1	TCP · 50	UDP · 55										
Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
2.21.35.219		10.0.0.2	443	2	180 bytes	11	4	248 bytes	0	0 bytes	20.968175	141.0000	14 bits/s	0 bits/s
4.252.185.49		10.0.0.2	443	7	709 bytes	0	7	709 bytes	0	0 bytes	0.000000	162.1943	34 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	12	248 bytes	9	12	248 bytes	0	0 bytes	9.540900	30.2275	530 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	2	180 bytes	14	2	180 bytes	0	0 bytes	160.883957	2.2091	651 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	2	558 bytes	8	9	558 bytes	0	0 bytes	8.581493	206.0798	21 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	2	168 bytes	5	2	168 bytes	0	0 bytes	6.163814	0.9569	1404 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	57	8 kB	13	33	6 kB	24	2 kB	54.405228	165.7563	308 bits/s	98 bits/s
10.0.0.2		10.0.0.2	443	10	620 bytes	6	10	620 bytes	0	0 bytes	7.543667	224.5504	22 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	10	700 bytes	10	10	700 bytes	0	0 bytes	10.175364	223.8944	25 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	30	2 kB	7	30	2 kB	0	0 bytes	7.569202	230.5041	56 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	6	454 bytes	3	6	454 bytes	0	0 bytes	5.963327	167.6308	21 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	4	256 bytes	4	4	256 bytes	0	0 bytes	5.963638	0.4230	4841 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	10	2 kB	12	10	2 kB	0	0 bytes	54.200163	163.9379	99 bits/s	0 bits/s
10.0.0.2		10.0.0.2	443	33,984	27 MB	1	12,779	3 MB	21,205	24 MB	0.407227	236.2275	95 kbps	806 kbps
10.0.0.2		10.0.0.2	443	14	1 kB	2	14	1 kB	0	0 bytes	0.707784	210.0444	47 bits/s	0 bits/s

I then proceeded to filter out these packets through taking from A ←→ Any.

ip.addr==2.21.35.219 && tcp.port==443					
No.	Time	Source	Destination	Protocol	Length Info
16	3.226917	2.21.35.219	10.0.0.2	TLSv1.2	78 Application Data
17	3.226958	10.0.0.2	2.21.35.219	TCP	54 54639 → 443 [ACK] Seq=1 Ack=25 Win=1024 Len=0
18	3.226974	2.21.35.219	10.0.0.2	TCP	62 443 → 54639 [FIN, ACK] Seq=25 Ack=1 Win=501 Len=0
19	3.226983	10.0.0.2	2.21.35.219	TCP	54 54639 → 443 [ACK] Seq=1 Ack=26 Win=1024 Len=0

It then presented me with this page which filtered out the packets. After this, I decided to look at HTTP protocol packets in particular

http					
No.	Time	Source	Destination	Protocol	Length Info
2120	49.485880	10.0.0.2	104.82.114.5	HTTP	323 GET /MFEwTzBNMEswSTA38gUrDgMCGuABBTjzY2p9Pa8o1bmj%2BNS%Wsz63kmHgQUuuhb2bU2FL3MdpdovdYxqTI%2
2122	49.489699	104.82.114.5	10.0.0.2	OCSP	1181 Response
11650	54.406539	10.0.0.2	10.0.0.4	HTTP	221 GET /dd.xml HTTP/1.1
11669	54.505514	10.0.0.4	10.0.0.2	HTTP/X..	482 HTTP/1.1 200 OK
26335	174.432217	10.0.0.2	10.0.0.4	HTTP	221 GET /dd.xml HTTP/1.1
26340	174.526289	10.0.0.4	10.0.0.2	HTTP/X..	62 HTTP/1.1 200 OK

▶ Frame 2120: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) on interface \Device\NPF_{430CE94F-F169-41A9-9813-97D64185C1} (00:00:00:00:00:00) on 0
 ▶ Ethernet II, Src: ASUSTekCOMPU_15:42:75 (f0:2f:74:15:42:75), Dst: Netgear_78:0d:d2 (a0:63:91:78:0d:d2)
 ▶ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 104.82.114.5
 ▶ Transmission Control Protocol, Src Port: 54664, Dst Port: 80, Seq: 1, Ack: 1, Len: 269
 ▶ Hypertext Transfer Protocol

0000	a0 63 91 78 0d d2 f0 2f 7
0010	01 35 26 57 40 00 80 06 0
0020	72 05 d5 88 00 50 a3 fb 6
0030	02 00 e5 80 00 00 47 45 5
0040	7a 42 4e 4d 45 73 77 53 5
0050	67 4d 43 47 67 55 41 42 4
0060	50 61 38 6f 69 62 6d 6a 2
0070	7a 36 33 6b 6d 57 67 51 5
0080	46 4c 33 4d 70 64 70 6f 7
0090	32 42 65 79 47 38 43 45 4
00a0	6e 25 32 42 56 68 38 62 3
00b0	25 33 44 20 48 54 54 50 2
00c0	63 68 65 2d 43 6f 6e 74 7
00d0	2d 61 67 65 20 3d 20 32 3
00e0	6e 65 63 74 69 6f 6e 3a 2
00f0	69 76 65 0d 0a 41 63 63 6
0100	0d 0a 55 73 65 72 2d 41 6
0110	63 72 6f 73 6f 66 74 2d 4
0120	49 2f 31 30 2e 30 0d 0a 4
0130	73 70 2e 64 69 67 69 63 6
0140	0a 0d 0a

This allows me to be able to see HTTPs and if I input this information into a compiler, I would then be able to view the website itself. Therefore if you open up a phishing email or send it to a dodgy website, through the use of wireshark, I am able to see the information of those packets and where they are being sent.

e \Device\NPF_{430CE94F-F169-41A9-9813-97D64185C1}	0000	a0 63 91 78 0d d2 f0 2f 7
00:63:91:78:0d:d2)	0010	01 35 26 57 40 00 80 06 0
9	0020	72 05 d5 88 00 50 a3 fb 6
	0030	02 00 e5 80 00 00 47 45 5
	0040	7a 42 4e 4d 45 73 77 53 5
	0050	67 4d 43 47 67 55 41 42 4
	0060	50 61 38 6f 69 62 6d 6a 2
	0070	7a 36 33 6b 6d 57 67 51 5
	0080	46 4c 33 4d 70 64 70 6f 7
	0090	32 42 65 79 47 38 43 45 4
	00a0	6e 25 32 42 56 68 38 62 3
	00b0	25 33 44 20 48 54 54 50 2
	00c0	63 68 65 2d 43 6f 6e 74 7
	00d0	2d 61 67 65 20 3d 20 32 3
	00e0	6e 65 63 74 69 6f 6e 3a 2
	00f0	69 76 65 0d 0a 41 63 63 6
	0100	0d 0a 55 73 65 72 2d 41 6
	0110	63 72 6f 73 6f 66 74 2d 4
	0120	49 2f 31 30 2e 30 0d 0a 4
	0130	73 70 2e 64 69 67 69 63 6
	0140	0a 0d 0a

We can also see here from the packets that the data is encrypted. A feature in which wireshark offers is that for hardcore security, I am able to upload a decryption key and directly see the data that is being sent in those packets.

15 0.445177	10.0.0.2	95.101.05.227	TCP	54 54900 → 445 [ACK] Seq=11904 ACK=451 Win=10
16 0.632416	Netgear_78:0d:d2	Broadcast	ARP	62 Who has 10.0.0.4? Tell 10.0.0.1
17 1.632447	Netgear_78:0d:d2	Broadcast	ARP	62 Who has 10.0.0.4? Tell 10.0.0.1
18 1.705413	143.200.200.10	10.0.0.2	UDP	74 443 → 54054 Len=33

From this picture, you can see ARP working in action. This means that my netgear router is trying to locate a device on my local network with the ip address 10.0.0.4, this is broadcasted on all of the devices on the subnet that receives this request. The router is trying to find the MAC address of 10.0.0.4 in order to deliver packets to it.

No.	Time	Source	Destination	Protocol	Length	Info
40	3.570976	10.0.0.2	74.125.193.94	TCP	55	50114 → 80 [ACK] Seq=1 Ack=1 Win=510 Len=1
41	3.583239	74.125.193.94	10.0.0.2	TCP	66	80 → 50114 [ACK] Seq=1 Ack=2 Win=1052 Len=0 SLE=1 SRE=2
3467	13.589734	10.0.0.2	74.125.193.94	TCP	55	[TCP Keep-Alive] 50114 → 80 [ACK] Seq=1 Ack=1 Win=510 Len=1
3468	13.600771	74.125.193.94	10.0.0.2	TCP	66	[TCP Keep-Alive ACK] 80 → 50114 [ACK] Seq=1 Ack=2 Win=1052 Len=0 SLE=1 SRE=2
3528	17.652033	10.0.0.2	104.82.114.5	TCP	66	50120 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
3562	17.680765	104.82.114.5	10.0.0.2	TCP	66	80 → 50120 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1412 SACK_PERM WS=128
3563	17.680798	10.0.0.2	104.82.114.5	TCP	54	50120 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
3564	17.680918	10.0.0.2	104.82.114.5	495		Request
3575	17.709925	104.82.114.5	10.0.0.2	OCSP	62	80 → 50120 [ACK] Seq=1 Ack=442 Win=64128 Len=0
3576	17.711039	104.82.114.5	10.0.0.2	OCSP	929	Response
3591	17.758459	10.0.0.2	104.82.114.5	TCP	54	50120 → 80 [ACK] Seq=442 Ack=876 Win=130304 Len=0
19919	19.630333	10.0.0.2	209.85.203.94	TCP	66	50134 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
19920	19.630428	10.0.0.2	209.85.203.94	TCP	66	50135 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
19921	19.641645	209.85.203.94	10.0.0.2	TCP	66	80 → 50134 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
19922	19.641700	10.0.0.2	209.85.203.94	TCP	54	50134 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
19923	19.641844	10.0.0.2	209.85.203.94	OCSP	504	Request
19924	19.643009	209.85.203.94	10.0.0.2	TCP	66	80 → 50135 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
19925	19.643031	10.0.0.2	209.85.203.94	TCP	54	50135 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
19926	19.643189	10.0.0.2	209.85.203.94	OCSP	503	Request
19928	19.653046	209.85.203.94	10.0.0.2	TCP	62	80 → 50134 [ACK] Seq=1 Ack=451 Win=269312 Len=0
19929	19.654069	209.85.203.94	10.0.0.2	TCP	62	80 → 50135 [ACK] Seq=1 Ack=450 Win=269312 Len=0
19931	19.670275	209.85.203.94	10.0.0.2	OCSP	756	Response
19935	19.673354	209.85.203.94	10.0.0.2	OCSP	755	Response
20030	19.714424	10.0.0.2	209.85.203.94	TCP	54	50135 → 80 [ACK] Seq=450 Ack=702 Win=130560 Len=0
20031	19.714458	10.0.0.2	209.85.203.94	TCP	54	50134 → 80 [ACK] Seq=451 Ack=703 Win=130560 Len=0
20106	19.741523	10.0.0.2	209.85.203.94	TCP	66	50137 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
20113	19.755141	209.85.203.94	10.0.0.2	TCP	66	80 → 50137 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256

I inputted a command that looked at port 80 using “Tcp.port == 80 command”, which scans all the packets from this port and presents them to me. Through this I was able to look at data that was being received and sent from this port.

<input checked="" type="checkbox"/>	Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack
<input checked="" type="checkbox"/>	HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/>	Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/>	OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/>	ICMP errors	icmp.type in { 3, 5, 11 } icmpv6.type in { 1, 4 }
<input checked="" type="checkbox"/>	ARP	arp
<input checked="" type="checkbox"/>	ICMP	icmp icmpv6
<input checked="" type="checkbox"/>	TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/>	SCPT ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/>	TTL low or unexpected	(ip.dst != 224.0.0.0/4 && ip.ttl < 5 && !ipm && !ospf) (ip.dst == 224.0.0.0/4 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp carp))
<input checked="" type="checkbox"/>	Checksum Errors	eth.fcs.status=="Bad" ip.checksum.status=="Bad" tcp.checksum.status=="Bad" udp.checksum.status=="Bad" sctp.checksum.status=="Bad" mstp.checksum.status=="Bad"
<input checked="" type="checkbox"/>	SMB	smb nbss nbns netbios
<input checked="" type="checkbox"/>	HTTP	http tcp.port == 80 http2
<input checked="" type="checkbox"/>	DCERPC	dcerpc
<input checked="" type="checkbox"/>	Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ismp
<input checked="" type="checkbox"/>	TCP SYN/FIN	tcp.flags & 0x02 tcp.flags.fin == 1
<input checked="" type="checkbox"/>	TCP	tcp
<input checked="" type="checkbox"/>	UDP	udp
<input checked="" type="checkbox"/>	Broadcast	eth[0] & 1
<input checked="" type="checkbox"/>	System Event	systemd_journal sysdig

In wireshark, certain packets are coloured in order for the user to understand more about the packets being sent as well as being able to easily identify any insecure packets that I may want to later investigate into.

Conclusion

In conclusion to my personal experiment, I tried to analyze my own home personal network using wireshark and nmap. When using these tools, I did not have a specific agenda or goal in mind, I was really only exploring around my network trying to learn more about the software application. Seeing how my pathway in computer science is cybersecurity, I strongly felt that networks is one of my most important modules and it presented to me an opportunity to be able to use these tools in real practice and present them. I also felt that nmap and wireshark would help provide me with a step-up in understanding more about networking and cybersecurity later down the line.

References:

<https://nmap.org/>

<https://www.wireshark.org/>

https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs

<https://www.freecodecamp.org/news/what-is-nmap-and-how-to-use-it-a-tutorial-for-the-greatest-scanning-tool-of-all-time/>

https://www.youtube.com/watch?v=wt_xMols4Ww&t=431s

<https://www.youtube.com/watch?v=qTaOZrDnMzQ>

<https://www.malware-traffic-analysis.net/>