Lab 4 Report

Alex Diaz, Hubert Meneses, Jonathan Ramos, Andres Ulloa

Contributions:

Alex: Recorded traffic, Ping flood attack, and nmap port scan through white shark on my own laptop, set up the slide structure, helped write structure of the report, designed and added images to both the report and the slides. I also kept the group up to date on lab progress as well as met multiple times on campus with group members in the classroom to be discuss and work on the project.

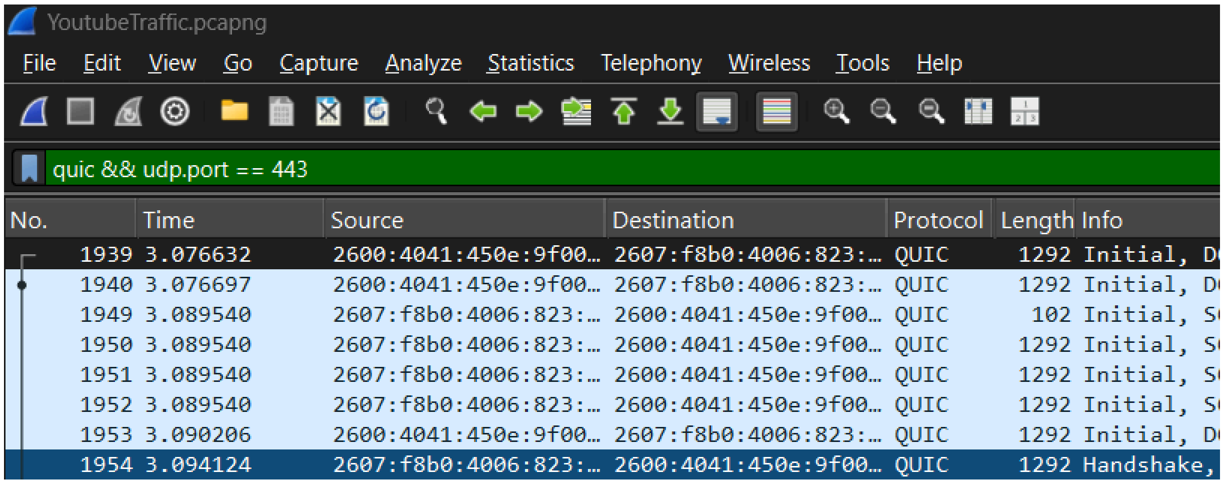
Andres: I worked on the report paragraph describing actions done during the lab and met up multiple times to edit the slides and work alongside the group to complete the project. Worked along with the group to cover any difficulties throughout the lab. Alex and I ran into some difficulty with task 3 which was using nmap since it was new for the both of us but managed to get it done.

Jonathan: Worked on goal 1 alongside with Alex to get the capture for YouTube with specific protocols. Referred the filters into wireshark so that we would get the correct IP's and data. Met multiple times with the group as well as calls to figure out some of the goals together and how we can accomplish them. Also chose YouTube as the application since originally hbo max not everyone had a login. Along with helping with some of the challenges faced since we had many we chose the ones that made sense to put into the slides.

Hubert: I worked on understanding the concept for the presentation . I helped form a little bit of the final report and double checked if the slides are easy to understand to the people we're presenting to. As well as checking if the group needs any additional help when meeting up with the group .

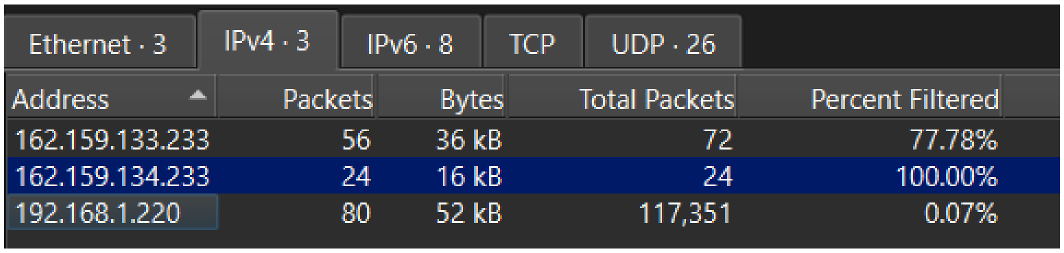
Our group worked on Lab 4: Network Monitoring and Security Awareness, this involved us starting something completely different that did not involve our network topology from the last two lab assignments. We were tasked with using wireshark to Capture network traffic using Wireshark and analyze IP addresses, MAC addresses, and protocols, Simulate a ping flood attack and observe network behavior, Detect port scanning attempts using nmap, and to Identify traffic anomalies such as high volume, unknown IPs, and unusual packet sizes. We first decided as a group to choose a website platform to monitor and we chose Youtube, it would enable us to see in real time packets going through wifi while the video is being played. Once we ran our test, we filtered the packets by using the command quic && udp.port == 443, from there we were able to see only how many packets were generated through Youtube's website specifically. To view the traffic IPs of our test, there is an endpoint section within wireshark that enables us to see all of the IPs that were being ran under that filter. While in the IPV4 tab, The filter showed at maximum 3 different IPs. Given the IPs, we wanted to find out where the IP addresses were coming from, so we went to a site that helped us explain in detail where the IPs were from. We used whois.domaintools.com, to integrate the IPs given and find their source. We found out that IP 163.159.133.233 & 162.159.134.233 were both coming from CloudFlare Inc. Going a little more in depth we found out that Cloudflare is very common to be used with Youtube for video deliveries, concluding that the packets shown for Youtube were being used within our tests. While we were running these tests within Wireshark, using the same statistics and endpoint settings, we were also able to see the MAC addresses that were being used. Given the IP addresses shown on the Ethernet tab, we got a maximum of three Addresses excluding (40:1a:58:e8:7c:68) being our own computers Wifi network adapter, which was used to stream the traffic. Moving forward within the same process, Mac addresses (b8:f8:53:87:2e:29 && b8:f8:53:87:2e:2b) shared the same manufacturer and we were able to find out that that it represented the different interfaces on our home router. Now this being mac addresses, we had to run a different source and we found maclookuo.app, being the perfect source to capture the where these addresses are coming from and confirm if there's any suspicious malicious encounters. Concluding Youtube traffic with Mac addresses, since the addresses only existed on the local network (layer 2), this proves to us that all the traffic captured moved between our device used to test and the router before it being forwarded to the external IP address. Moving forward with stimulating the flood ping attack, we ran into some difficulties being that we were unsure on how to correctly showcase our flood ping attack that occurred during our test. As we started with the task, we also found it difficult to ping flood attack while running a video using Youtube. So, we decided to run our general network and stimulated it that way, and found the results more pleasing. The way we came upon with the task, we used and ran command ping -t 192.168.1.1 within the command prompt and at the same time Wireshark packet tracer was on and running. As we ran everything all at once, we saw in real time a lot of ICMP echo requests being sent across the network. The packets appeared at a very high rate and it looks like it intentionally generated traffic where continuous ICMP traffic would swamp a network if ran for too long. Using filter imcp.type == 8, we found it more efficient to only show ICMP echo requests packets done through the ping flood attack. Heading over to detecting port scanning attempts using Nmap. Us as a group it was our first time using this program, so we had to do some research on commands and install Nmap itself. We then later proceeded to use it in order to benefit our task. We performed a SYN port scan using Nmap while having Wireshark capture the packets. Command nmap -sS <our IP> was used to capture the open ports within the network. Large amounts of TCP SYP packets were displayed and sent to different port numbers, This is proved to be used nowadays for port scamming attempts.

Packet Screenshots Captured


YoutubeTraffic.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

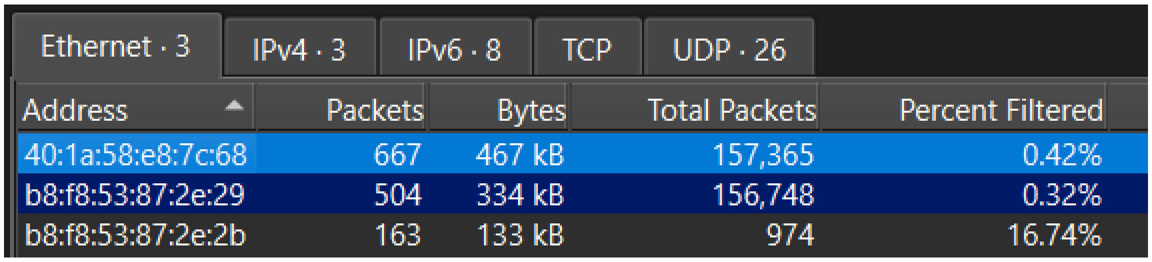
```
quic && udp.port == 443
```

No.	Time	Source	Destination	Protocol	Length	Info
1939	3.076632	2600:4041:450e:9f00…	2607:f8b0:4006:823:…	QUIC	1292	Initial, D
1940	3.076697	2600:4041:450e:9f00…	2607:f8b0:4006:823:…	QUIC	1292	Initial, D
1949	3.089540	2607:f8b0:4006:823:…	2600:4041:450e:9f00…	QUIC	102	Initial, S
1950	3.089540	2607:f8b0:4006:823:…	2600:4041:450e:9f00…	QUIC	1292	Initial, S
1951	3.089540	2607:f8b0:4006:823:…	2600:4041:450e:9f00…	QUIC	1292	Initial, S
1952	3.089540	2607:f8b0:4006:823:…	2600:4041:450e:9f00…	QUIC	1292	Initial, S
1953	3.090206	2600:4041:450e:9f00…	2607:f8b0:4006:823:…	QUIC	1292	Initial, D
1954	3.094124	2607:f8b0:4006:823:…	2600:4041:450e:9f00…	QUIC	1292	Handshake,



Ethernet · 3 | IPv4 · 3 | IPv6 · 8 | TCP | UDP · 26

Address	Packets	Bytes	Total Packets	Percent Filtered
162.159.133.233	56	36 kB	72	77.78%
162.159.134.233	24	16 kB	24	100.00%
192.168.1.220	80	52 kB	117,351	0.07%

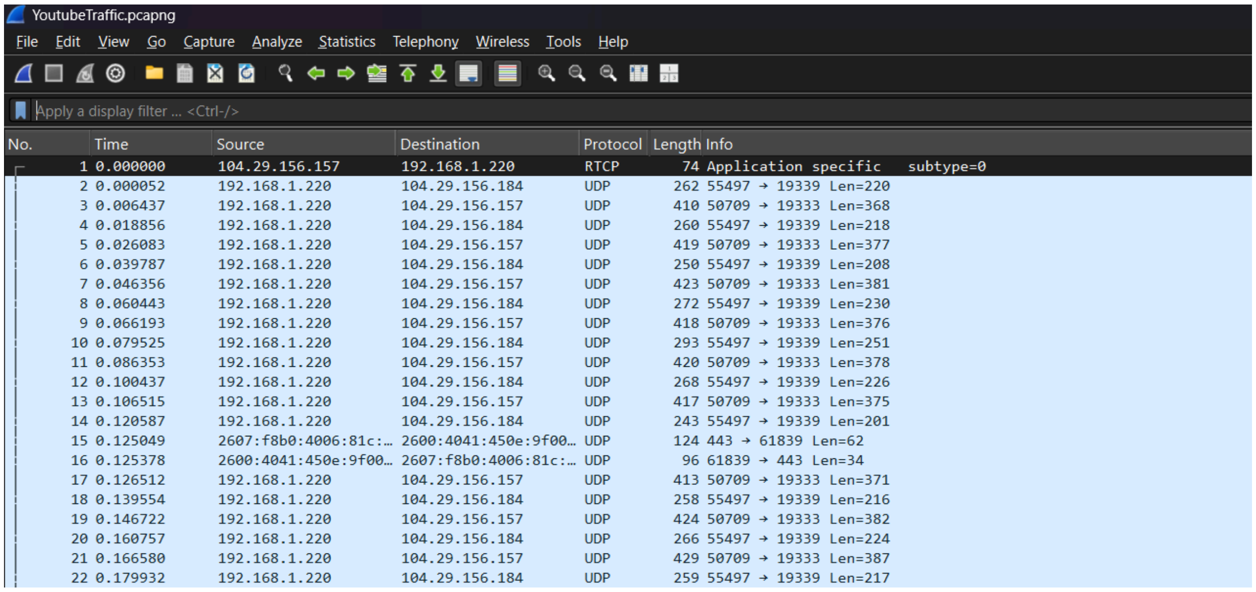


Ethernet · 3 | IPv4 · 3 | IPv6 · 8 | TCP | UDP · 26

Address	Packets	Bytes	Total Packets	Percent Filtered
40:1a:58:e8:7c:68	667	467 kB	157,365	0.42%
b8:f8:53:87:2e:29	504	334 kB	156,748	0.32%
b8:f8:53:87:2e:2b	163	133 kB	974	16.74%

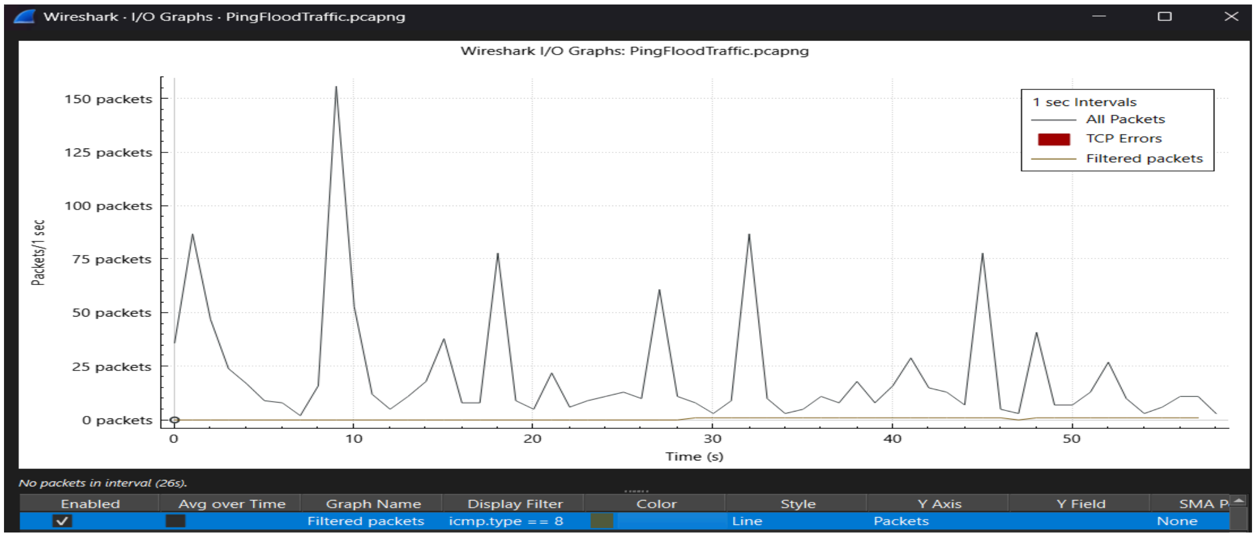
Normal vs Suspicious Traffic

Normal Traffic


YoutubeTraffic.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

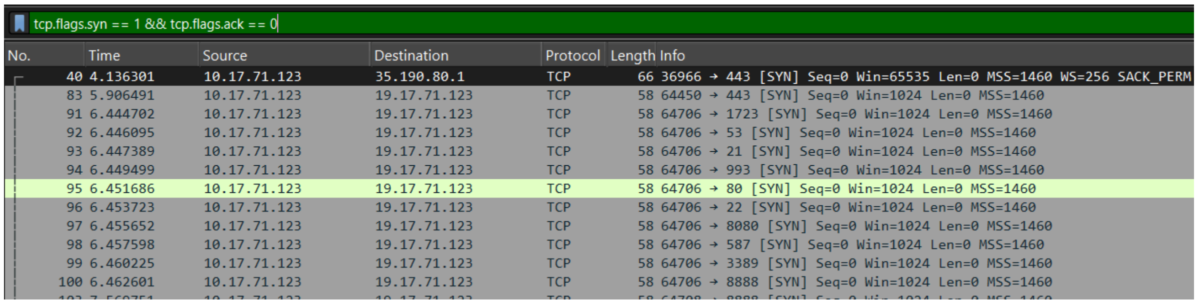
```
No.   Time      Source             Destination        Protocol Length Info
    1 0.000000  104.29.156.157     192.168.1.220      RTCP       74 Application specific   subtype=0
    2 0.000052  192.168.1.220      104.29.156.184     UDP       262 55497 → 19339 Len=220
    3 0.006437  192.168.1.220      104.29.156.157     UDP       410 50709 → 19333 Len=368
    4 0.018856  192.168.1.220      104.29.156.184     UDP       260 55497 → 19339 Len=218
    5 0.026083  192.168.1.220      104.29.156.157     UDP       419 50709 → 19333 Len=377
    6 0.039787  192.168.1.220      104.29.156.184     UDP       250 55497 → 19339 Len=208
    7 0.046356  192.168.1.220      104.29.156.157     UDP       423 50709 → 19333 Len=381
    8 0.060443  192.168.1.220      104.29.156.184     UDP       272 55497 → 19339 Len=230
    9 0.066193  192.168.1.220      104.29.156.157     UDP       418 50709 → 19333 Len=376
   10 0.079525  192.168.1.220      104.29.156.184     UDP       293 55497 → 19339 Len=251
   11 0.086353  192.168.1.220      104.29.156.157     UDP       420 50709 → 19333 Len=378
   12 0.100437  192.168.1.220      104.29.156.184     UDP       268 55497 → 19339 Len=226
   13 0.106515  192.168.1.220      104.29.156.157     UDP       417 50709 → 19333 Len=375
   14 0.120587  192.168.1.220      104.29.156.184     UDP       243 55497 → 19339 Len=201
   15 0.125049  2607:f8b0:4006:81c:… 2600:4041:450e:9f00… UDP  124 443 → 61839 Len=62
   16 0.125378  2600:4041:450e:9f00… 2607:f8b0:4006:81c:… UDP   96 61839 → 443 Len=34
   17 0.126512  192.168.1.220      104.29.156.157     UDP       413 50709 → 19333 Len=371
   18 0.139554  192.168.1.220      104.29.156.184     UDP       258 55497 → 19339 Len=216
   19 0.146722  192.168.1.220      104.29.156.157     UDP       424 50709 → 19333 Len=382
   20 0.160757  192.168.1.220      104.29.156.184     UDP       266 55497 → 19339 Len=224
   21 0.166580  192.168.1.220      104.29.156.157     UDP       429 50709 → 19333 Len=387
   22 0.179932  192.168.1.220      104.29.156.184     UDP       259 55497 → 19339 Len=217
```

Suspicious Traffic



No packets in interval (26s).

Enabled	Avg over Time	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA P
✓		Filtered packets	icmp.type == 8		Line	Packets		None



```
tcp.flags.syn == 1 && tcp.flags.ack == 0

No.   Time      Source        Destination   Protocol Length Info
   40 4.136301  10.17.71.123  35.190.80.1   TCP      66 36966 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   83 5.906491  10.17.71.123  19.17.71.123  TCP      58 64450 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   91 6.444702  10.17.71.123  19.17.71.123  TCP      58 64706 → 1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   92 6.446095  10.17.71.123  19.17.71.123  TCP      58 64706 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   93 6.447389  10.17.71.123  19.17.71.123  TCP      58 64706 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   94 6.449499  10.17.71.123  19.17.71.123  TCP      58 64706 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   95 6.451686  10.17.71.123  19.17.71.123  TCP      58 64706 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   96 6.453723  10.17.71.123  19.17.71.123  TCP      58 64706 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   97 6.455652  10.17.71.123  19.17.71.123  TCP      58 64706 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   98 6.457598  10.17.71.123  19.17.71.123  TCP      58 64706 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   99 6.460225  10.17.71.123  19.17.71.123  TCP      58 64706 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
  100 6.462601  10.17.71.123  19.17.71.123  TCP      58 64706 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

Summary Of Security Threats Identified

During the YouTube traffic capture, the packets appeared normal in terms of regular web streaming. The protocols observed included QUIC, TLS, TCP, and UDP, all flowing normally between our device and the Google/Cloudflare servers. Packet sizes and timing were stable, indicating to us normal browsing and video playback behavior.

The IPs we observed were the IPs **162.159.133.233** and **162.159.134.233**, they were initially unknown, but through using a WHOIS lookup tool, we identified both addresses as belonging to Cloudflare which gave us the conclusion that there were no malicious anomalies here..

In contrast, the ping flood traffic showed clear anomalies. Wireshark displayed hundreds of ICMP Echo Request packets, and the I/O graph showed high volume with large spikes in packets per second, which is not normal in usual traffic.

With Nmap, we observed bursts of TCP SYN packets being sent to many different port numbers rapidly.

Some ports responded with SYN/ACK (open), while most returned RST (closed).

By comparing these captures, normal traffic has steady, packet flow with consistent protocols, while non normal traffic shows either high volume (ping flood) or a wide range of targeted ports (port scan). These anomalies highlight how Wireshark can be used to see how routine network usage can potentially be malicious activity.