ISCS 3523-004

Lab 02: Event Analysis

Christian Barba; fgb587

October 9th, 2022

Table of Contents

Introduction

The purpose of this lab is to examine and analyze a .pcap file using the various resources given to us through our virtual machines. By dissecting the file, we will be able to determine a lot of information regarding the capture and determine if this was malicious or not. The owner of the file claims that they use the internet for emails, browsing, and to play video games, so these activities will be expected to appear on the capture. However, this does not rule out menacing activity as a hacker could be trying to cover their tracks through these activities as well.

Questions A & B

In order to start breaking down what has been presented to us, we must first start with the basic information: how many packets did Wireshark capture and how long did the session last? When opening the file in Wireshark, it is listed in order of each packet starting with the first one. By inspecting the frame of the first packet, we can see in figure 1.1 the time stamp in which this packet arrived with it being on September 12, 2022, 21:32:58 MDT. By choosing to go to the last packet, shown by figure 1.2, we can see that the date stayed the same and the time was 21:55:59. This can tell us that the session lasted about 23 minutes. By scrolling to the end, we can also see that the last packet was also numbered 33,788. Another way I found to solve this problem is by clicking on the tab "Capture File Properties". This will show the time of the first packet and the last packet as well as the total number of packets captured.

Questions C & D

The tab "Capture File Properties" as stated previously will also display the information about the total bytes. The total bytes captured was 253,324,419. This would make the total session worth about 253 megabytes in total. Now trying to find the number of protocols at use

required me to dive into the protocol hierarchy. Figure 2.1 shows a complete list of all of the protocols that are being used in a hierarchal sense. The three main protocols that split off into sub protocols, except ARP, are the Internet Protocol Version 6 (IPv6), Internet Protocol Version 4 (IPv4), and Address Resolution Protocol (ARP). IPv6 is home to the UDP, MDNS, and LLMNR sublevel protocols. IPv4 also has sublevels of UDP, SSDP, QUIC IETF, SMB, BROWSER, MDNS, LLMNR, DHCP, DNS, TCP, TLS, HTP, IGMP, and ICMP protocols.

Questions E & F

I am now in need of trying to find out when the bulk of the data got transmitted. When meddling with Wireshark, I went through the statistics tab to see if I could find anything that would allow me to see a timeline of the packets that were transmitted into the computer. Luckily, when inspecting the I/O Graph tab, I was able to see all of the traffic mapped out onto a single graph shown in figure 3.1. The I/O graph is responsible for displaying packet statistics based on the filter you add or subtract to the line graph. I chose to display all packets, and this was able to show me that the bulk of the data that was transmitted both in and out from the computer happened at around the 7 second mark with spikes occurring at packets 16s, 20s, 46s, and 90s. There also seems to have been another cluster at packet 862s with spikes occurring at packets 871s, 873s, and 938s.

The transmission seems to have spiked due to a surplus of TCP based packets were being exchange between the host computer and a couple of other sources as shown in figure 3.2. This led me to believe that the host computer might have been in the middle of a download during these spikes because of the number of packets that were able to be received by the system. Some of these packets also appear to be rejected, and I am also seeing that port 443 was involved in the dropping of certain packets, which may be the reason why port 80 was accessed. This did not

raise any concerns for a DDoS attack because I could see that the host computer was still able to function properly after these spikes in transmission.

Questions H & I

In order to find the name of the host computer, I ran the .pcap file through Network Miner on my Kali machine. From here, I clicked on the "hosts" tab to see if I could run into any IP addresses that had the word "local" attached to it. I thought that this might be a major indication of finding the actual host computer. I also went looking with the knowledge that the IP address for the machine might be 172.16.2.4, since this was the address that transmitted the first packet on Wireshark. To no surprise, I was able to find the host with the address 172.16.2.4 and I was able to view a lot of information about the machine as shown in figure 4.1. The name of the machine was "DESKTOP-1DNMB8Q". From viewing the host on Network Miner, I was also able to what operating system the host was using. The host was using Microsoft Windows.

Questions J & K

Since I knew that the host IP address was 172.16.2.4, I knew I could use Network Miner to find out the rest of the local area topology. Doing this would not be that difficult because I knew that the local area devices would be attached to 172.16 local sub network. The only other three IP addresses I could find with the same opening two bytes as the local network were the default gateway (172.16.2.1), the NIC Vendor: Broadcast (172.16.2.255), and finally a host with the name 172.16.2.2. The first two I could rule that they were helping the host computer. The default gateway acts as a router that moves packets around to their desired destination. The broadcast IP address is "used to transmit data to all of the hosts on the local subnet" (*Broadcast Address*). This leaves us with the final IP address of 172.16.2.2, but not much information is shown to help indicate what this could be. The OS is unknown, but the hop distance is equal to 0,

so there can be an assumption that 172.16.2.2 IP address could be attached to the host computer as well. A key detail that is shown however is that port 80 is open. This could be an indicator that something unencrypted could have traveled to this device, since there have been 210 packets sent and 164 packets received over this device. Other than these IP addresses, the local network seems to have nothing else attached to it. These devices are also nameless compared to the actual host computer.

Questions L & M

To try and figure out if there had been an attempt to log into any other computer on the network through the host computer, I would have to look in the "Credentials" tab in Network Miner. This tab shows any log on attempts that would have occurred throughout this capture, but there were no listings here. When looking at the "Sessions" tab however, I was able to see that there was communication between the host computer and 172.16.2.2. What I can conclude form this is that the host computer reached out, and/or was contacted through port 80 in order to communicate with unverified sites. Other than this, there seems to be no other incidents of the host accessing any other computers on the local network. Through this, I was able to conclude that besides 172.16.2.2, which is used to communicate with port 80, there were no other devices or services on the host computer.

The Story the Capture Tells

After analyzing the file thoroughly, I concluded that while browsing the internet, the owner of the host computer came across a way to download some files, and the files ended up being malicious. By running Network miner in my Windows machine, I was able to go to the files that were downloaded and interact with them. The files listed were a "GTAVUpdate.exe"

file, an “index.html” file, a “favicon.ico.html” file, a “secret.html” file, and an “openlogo-75.png” file. Going back to Wireshark shows that the files were all downloaded using the “get” command through TCP port 80 as shown in figure 5.1. After seeing when the files were downloaded, I tried to open the GTA V update application. An error, as displayed in figure 5.2, shows that there was a virus located within the .exe file that I tried to open.

I now know that the host computer is most likely infected if the .exe file had been run at all. Another indicator that these files might be malicious is that they used port 80 to be transferred. This is an unencrypted port that does not need secure websites to transfer data. Seeing that the downloaded files came from here is a pretty big indication that something malicious could be going on, but this port is also used to help redirect traffic from HTTP to HTTPS to add the level of security that it needs, so we can’t assume everything operating through this port is malicious (Seaton). Gathering all of this data is a clear indicator of poor operation security on the hosts part because there is no need to download risky file from untrusted HTTP sources when the official files are available from trusted servers.

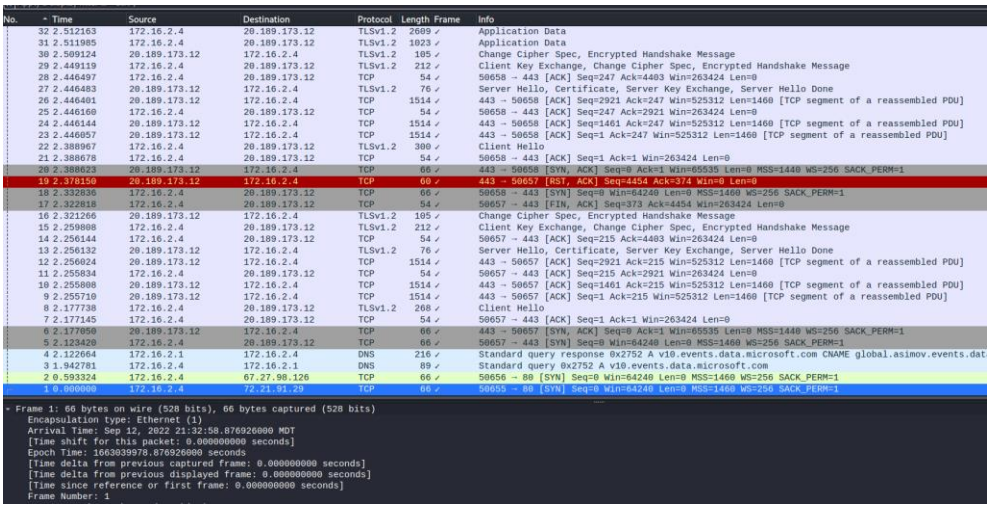
Suricata

In order to run the file on Suricata, I needed to secure copy the file from my Kali machine to my IPS machine using the command “scp Lab2.pcap cbarba@172.16.1.4:/home/cbarba”. Once this was complete, I used a command to have Suricata read the pcap file and determine if there were alerts that were activated. In figure 5.2, it is evident that Suricata was able to read the file and determine that 25730 alerts were triggered. This was enough for me to double down on the conclusion that the downloaded files were malicious. By looking into the eve.json file, it is evident that a lot of the alerts were TCP and UDP related. The main issue I saw plaguing the alerts was an invalid checksum which drops bad packets that do not get acknowledge when the

three-way handshake is occurring (*Packet data stripped by "TCP Invalid Checksum" IPS protection)*. This could be due to a surplus of bad traffic happening on the system.

Appendix

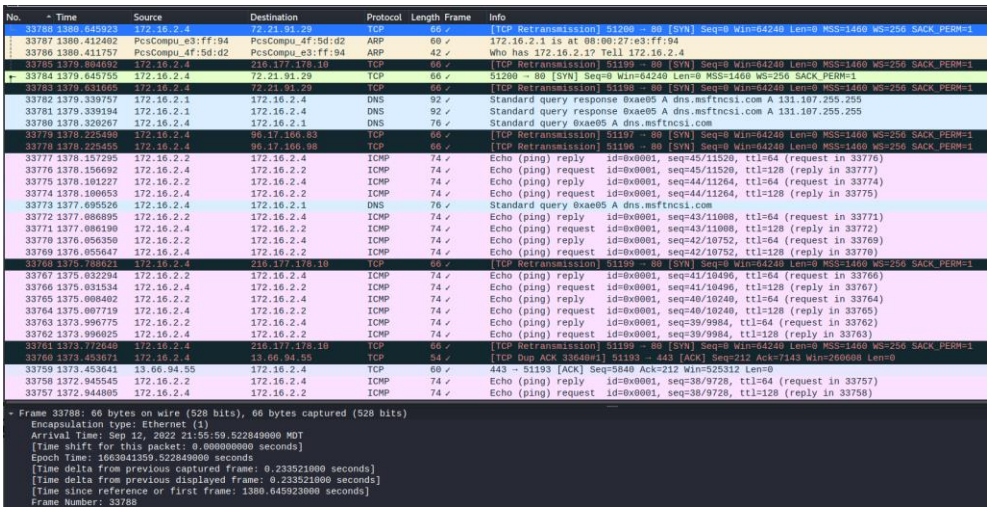
Figure 1.1 – Start of the capture



No.	Time	Source	Destination	Protocol	Length	Frame	Info
32	2.512163	172.16.2.4	20.189.173.12	TLSv1.2	2609	✓	Application Data
31	2.511985	172.16.2.4	20.189.173.12	TLSv1.2	1023	✓	Application Data
30	2.509124	20.189.173.12	172.16.2.4	TLSv1.2	105	✓	Change Cipher Spec, Encrypted Handshake Message
29	2.449119	172.16.2.4	20.189.173.12	TLSv1.2	212	✓	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
28	2.446497	172.16.2.4	20.189.173.12	TCP	54	✓	50658 → 443 [ACK] Seq=247 Ack=4403 Win=263424 Len=0
27	2.446483	20.189.173.12	172.16.2.4	TLSv1.2	76	✓	Server Hello, Certificate, Server Key Exchange, Server Hello Done
26	2.446401	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50658 [ACK] Seq=2921 Ack=247 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
25	2.446160	172.16.2.4	20.189.173.12	TCP	54	✓	50658 → 443 [ACK] Seq=247 Ack=2921 Win=263424 Len=0
24	2.446144	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50658 [ACK] Seq=1461 Ack=247 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
23	2.446057	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50658 [ACK] Seq=1 Ack=247 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
22	2.380967	172.16.2.4	20.189.173.12	TLSv1.2	300	✓	Client Hello
21	2.380678	172.16.2.4	20.189.173.12	TCP	54	✓	50658 → 443 [ACK] Seq=1 Ack=1 Win=263424 Len=0
20	2.380623	20.189.173.12	172.16.2.4	TCP	66	✓	443 → 50658 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
19	2.378150	20.189.173.12	172.16.2.4	TCP	60	✓	443 → 50657 [RST, ACK] Seq=4454 Ack=374 Win=0 Len=0
18	2.332836	172.16.2.4	20.189.173.12	TCP	66	✓	50658 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
17	2.322818	172.16.2.4	20.189.173.12	TCP	54	✓	50657 → 443 [FIN, ACK] Seq=373 Ack=4454 Win=263424 Len=0
16	2.321266	20.189.173.12	172.16.2.4	TLSv1.2	105	✓	Change Cipher Spec, Encrypted Handshake Message
15	2.259008	172.16.2.4	20.189.173.12	TLSv1.2	212	✓	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	2.256144	172.16.2.4	20.189.173.12	TCP	54	✓	50657 → 443 [ACK] Seq=215 Ack=4403 Win=263424 Len=0
13	2.256132	20.189.173.12	172.16.2.4	TLSv1.2	76	✓	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	2.256024	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50657 [ACK] Seq=2921 Ack=215 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
11	2.255834	172.16.2.4	20.189.173.12	TCP	54	✓	50657 → 443 [ACK] Seq=215 Ack=2921 Win=263424 Len=0
10	2.255808	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50657 [ACK] Seq=1461 Ack=215 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
9	2.255710	20.189.173.12	172.16.2.4	TCP	1514	✓	443 → 50657 [ACK] Seq=1 Ack=215 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
8	2.177738	172.16.2.4	20.189.173.12	TLSv1.2	268	✓	Client Hello
7	2.177145	172.16.2.4	20.189.173.12	TCP	54	✓	50657 → 443 [ACK] Seq=1 Ack=1 Win=263424 Len=0
6	2.177050	20.189.173.12	172.16.2.4	TCP	66	✓	443 → 50657 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
5	2.123420	172.16.2.4	20.189.173.12	TCP	66	✓	50657 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	2.122664	172.16.2.1	172.16.2.4	DNS	216	✓	Standard query response 0x2752 A v10.events.data.microsoft.com CNAME global.asimov.events.dat
3	1.942781	172.16.2.4	172.16.2.1	DNS	89	✓	Standard query 0x2752 A v10.events.data.microsoft.com
2	0.593324	172.16.2.4	67.27.98.126	TCP	66	✓	50656 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1	0.000000	172.16.2.4	72.21.91.29	TCP	66	✓	50655 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

```
▾ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 12, 2022 21:32:58.876926000 MDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1663039978.876926000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
```

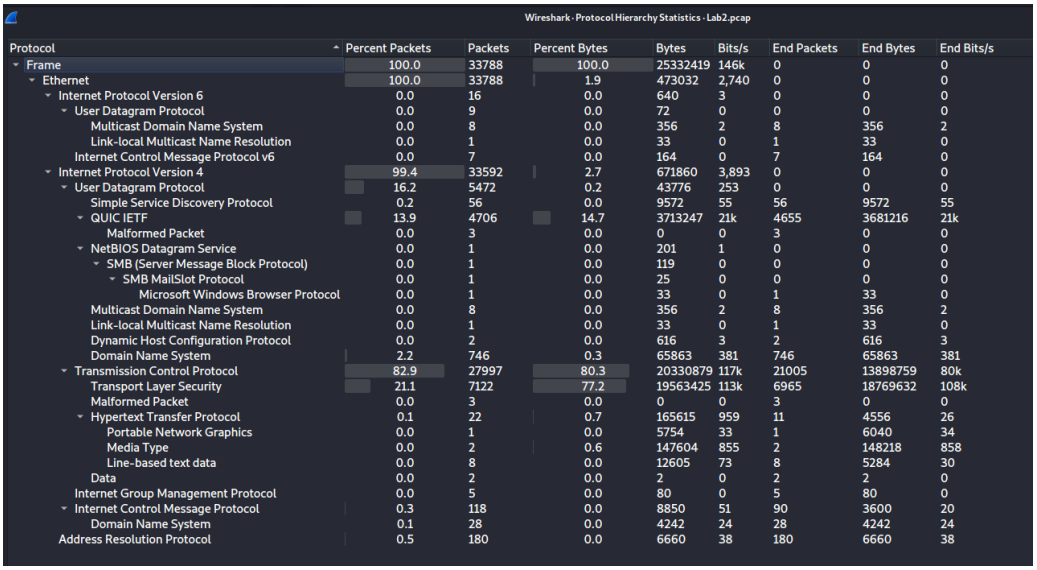
Figure 1.2 – End of the capture



No.	Time	Source	Destination	Protocol	Length	Frame	Info
33788	1380.645923	172.16.2.4	72.21.91.29	TCP	66	✓	[TCP Retransmission] 51200 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33787	1380.412402	PcsCompu_e3:ff:94	PcsCompu_4f:5d:d2	ARP	60	✓	172.16.2.1 is at 08:00:27:e3:ff:94
33786	1380.411757	PcsCompu_4f:5d:d2	PcsCompu_e3:ff:94	ARP	42	✓	Who has 172.16.2.1? Tell 172.16.2.4
33785	1379.884692	172.16.2.4	216.177.178.10	TCP	66	✓	[TCP Retransmission] 51199 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33784	1379.645755	172.16.2.4	72.21.91.29	TCP	66	✓	51200 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33783	1379.631665	172.16.2.4	72.21.91.29	TCP	66	✓	[TCP Retransmission] 51198 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33782	1379.339757	172.16.2.1	172.16.2.4	DNS	92	✓	Standard query response 0xae05 A dns.msftncsi.com A 131.107.255.255
33781	1379.339194	172.16.2.1	172.16.2.4	DNS	92	✓	Standard query response 0xae05 A dns.msftncsi.com A 131.107.255.255
33780	1378.320267	172.16.2.4	172.16.2.1	DNS	76	✓	Standard query 0xae05 A dns.msftncsi.com
33779	1378.225490	172.16.2.4	96.17.166.83	TCP	66	✓	[TCP Retransmission] 51197 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33778	1378.225455	172.16.2.4	96.17.166.98	TCP	66	✓	[TCP Retransmission] 51196 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33777	1378.157295	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=45/11520, ttl=64 (request in 33776)
33776	1378.156692	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=45/11520, ttl=128 (reply in 33777)
33775	1378.101227	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=44/11264, ttl=64 (request in 33774)
33774	1378.100653	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=44/11264, ttl=128 (reply in 33775)
33773	1377.695526	172.16.2.4	172.16.2.1	DNS	76	✓	Standard query 0xae05 A dns.msftncsi.com
33772	1377.086895	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=43/11008, ttl=64 (request in 33771)
33771	1377.086190	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=43/11008, ttl=128 (reply in 33772)
33770	1376.056350	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=42/10752, ttl=64 (request in 33769)
33769	1376.055647	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=42/10752, ttl=128 (reply in 33770)
33768	1375.788621	172.16.2.4	216.177.178.10	TCP	66	✓	[TCP Retransmission] 51199 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33767	1375.032294	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=41/10496, ttl=64 (request in 33766)
33766	1375.031534	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=41/10496, ttl=128 (reply in 33767)
33765	1375.008402	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=40/10240, ttl=64 (request in 33764)
33764	1375.007719	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=40/10240, ttl=128 (reply in 33765)
33763	1373.996775	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=39/9984, ttl=64 (request in 33762)
33762	1373.996025	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=39/9984, ttl=128 (reply in 33763)
33761	1373.772648	172.16.2.4	216.177.178.10	TCP	66	✓	[TCP Retransmission] 51199 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33760	1373.453671	172.16.2.4	13.66.94.55	TCP	54	✓	[TCP Dup ACK 33640#1] 51193 → 443 [ACK] Seq=213 Ack=7143 Win=260608 Len=0
33759	1373.453641	13.66.94.55	172.16.2.4	TCP	60	✓	443 → 51193 [ACK] Seq=5840 Ack=212 Win=525312 Len=0
33758	1372.945545	172.16.2.2	172.16.2.4	ICMP	74	✓	Echo (ping) reply id=0x0001, seq=38/9728, ttl=64 (request in 33757)
33757	1372.944805	172.16.2.4	172.16.2.2	ICMP	74	✓	Echo (ping) request id=0x0001, seq=38/9728, ttl=128 (reply in 33758)

```
▾ Frame 33788: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 12, 2022 21:55:59.522849000 MDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1663041359.522849000 seconds
    [Time delta from previous captured frame: 0.233521000 seconds]
    [Time delta from previous displayed frame: 0.233521000 seconds]
    [Time since reference or first frame: 1380.645923000 seconds]
    Frame Number: 33788
```

Figure 2.1 – Protocol Hierarchy



Wireshark · Protocol Hierarchy Statistics · Lab2.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▾ Frame	100.0	33788	100.0	25332419	146k	0	0	0
▾ Ethernet	100.0	33788	1.9	473032	2,740	0	0	0
▾ Internet Protocol Version 6	0.0	16	0.0	640	3	0	0	0
▾ User Datagram Protocol	0.0	9	0.0	72	0	0	0	0
Multicast Domain Name System	0.0	8	0.0	356	2	8	356	2
Link-local Multicast Name Resolution	0.0	1	0.0	33	0	1	33	0
Internet Control Message Protocol v6	0.0	7	0.0	164	0	7	164	0
▾ Internet Protocol Version 4	99.4	33592	2.7	671860	3,893	0	0	0
▾ User Datagram Protocol	16.2	5472	0.2	43776	253	0	0	0
Simple Service Discovery Protocol	0.2	56	0.0	9572	55	56	9572	55
▾ QUIC IETF	13.9	4706	14.7	3713247	21k	4655	3681216	21k
Malformed Packet	0.0	3	0.0	0	0	3	0	0
▾ NetBIOS Datagram Service	0.0	1	0.0	201	1	0	0	0
▾ SMB (Server Message Block Protocol)	0.0	1	0.0	119	0	0	0	0
▾ SMB MailSlot Protocol	0.0	1	0.0	25	0	0	0	0
Microsoft Windows Browser Protocol	0.0	1	0.0	33	0	1	33	0
Multicast Domain Name System	0.0	8	0.0	356	2	8	356	2
Link-local Multicast Name Resolution	0.0	1	0.0	33	0	1	33	0
Dynamic Host Configuration Protocol	0.0	2	0.0	616	3	2	616	3
Domain Name System	2.2	746	0.3	65863	381	746	65863	381
▾ Transmission Control Protocol	82.9	27997	80.3	20330879	117k	21005	13898759	80k
Transport Layer Security	21.1	7122	77.2	19563425	113k	6965	18769632	108k
Malformed Packet	0.0	3	0.0	0	0	3	0	0
▾ Hypertext Transfer Protocol	0.1	22	0.7	165615	959	11	4556	26
Portable Network Graphics	0.0	1	0.0	5754	33	1	6040	34
Media Type	0.0	2	0.6	147604	855	2	148218	858
Line-based text data	0.0	8	0.0	12605	73	8	5284	30
Data	0.0	2	0.0	2	0	2	2	0
Internet Group Management Protocol	0.0	5	0.0	80	0	5	80	0
▾ Internet Control Message Protocol	0.3	118	0.0	8850	51	90	3600	20
Domain Name System	0.1	28	0.0	4242	24	28	4242	24
Address Resolution Protocol	0.5	180	0.0	6660	38	180	6660	38

Figure 3.1 – I/O Graph from Wireshark

Figure 3.2 – Transmission Spikes

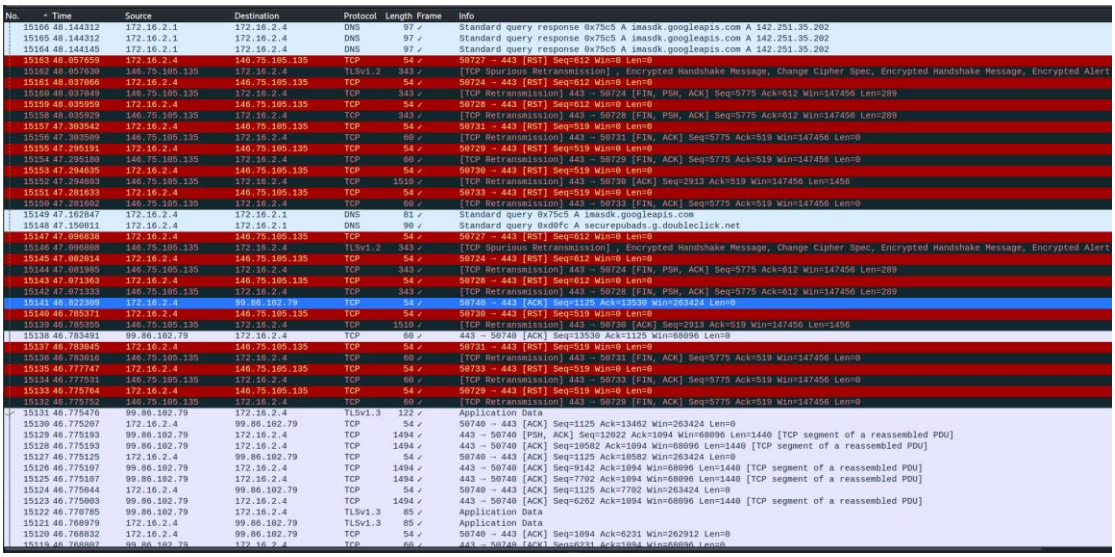


Figure 4.1 – Network Miner Host Computer

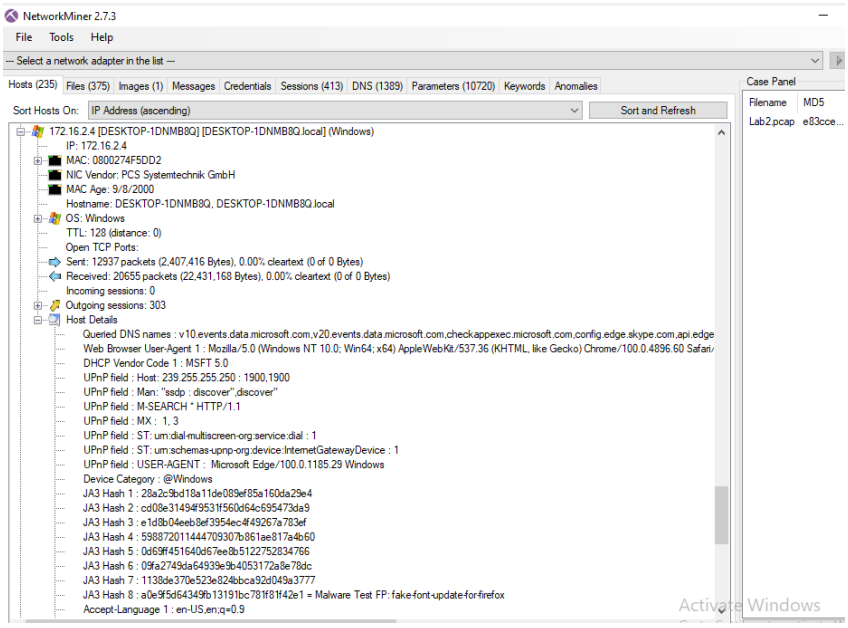
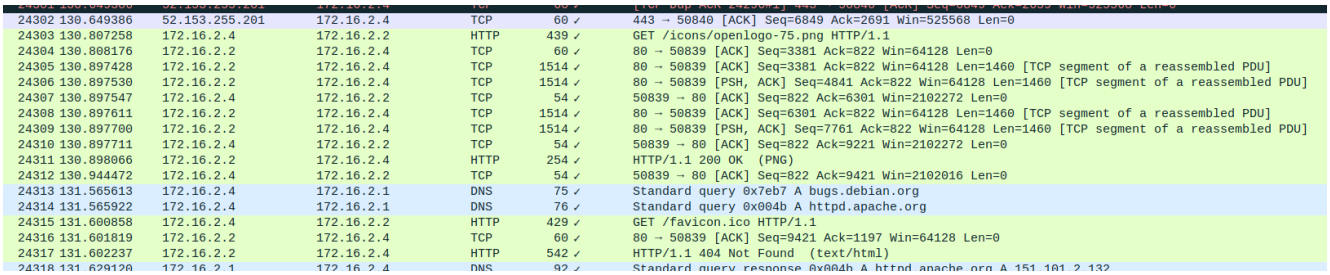
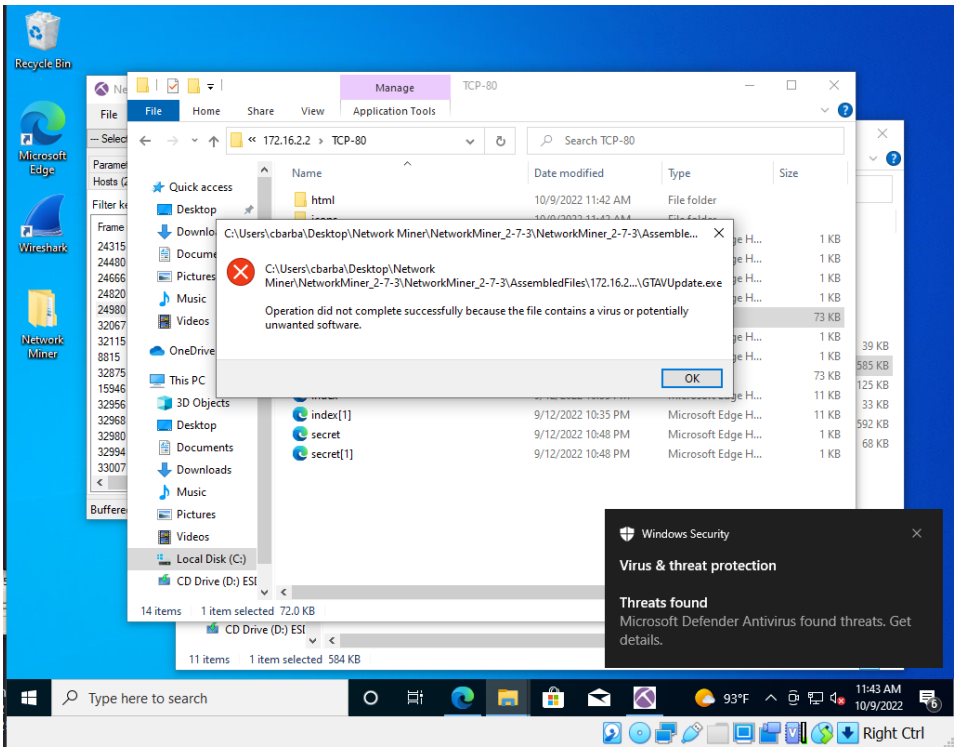


Figure 5.1 – “get” Command in Action

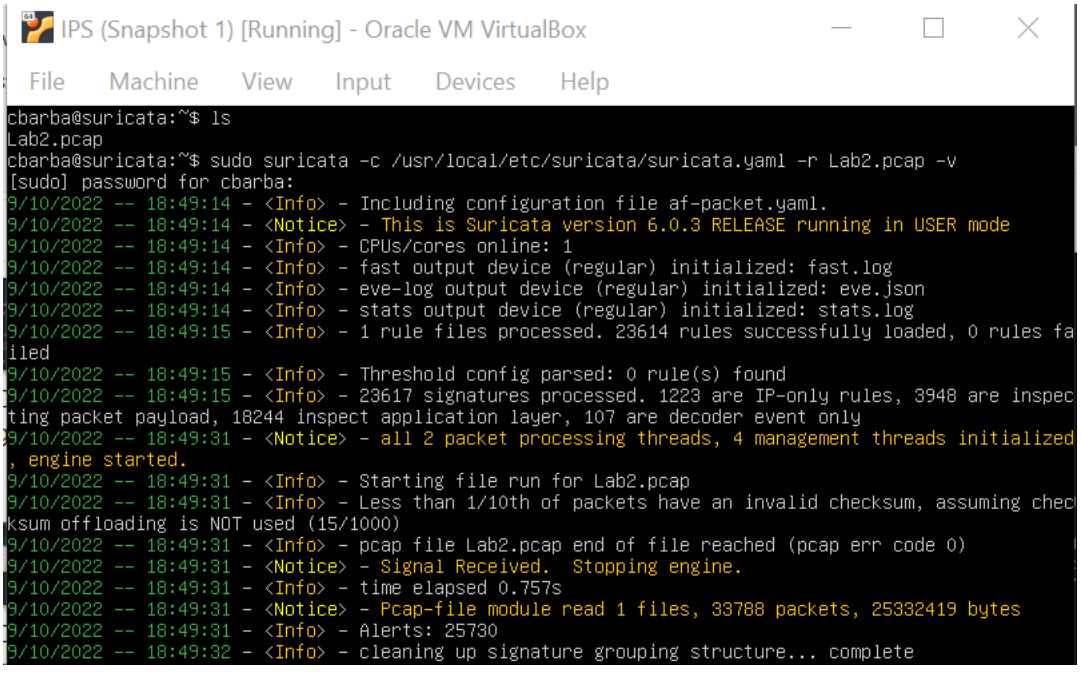


24302	130.649386	52.153.255.201	172.16.2.4	TCP	60 ✓	443 → 50840 [ACK] Seq=6849 Ack=2691 Win=525568 Len=0
24303	130.807258	172.16.2.4	172.16.2.2	HTTP	439 ✓	GET /icons/openlogo-75.png HTTP/1.1
24304	130.808176	172.16.2.2	172.16.2.4	TCP	60 ✓	80 → 50839 [ACK] Seq=3381 Ack=822 Win=64128 Len=0
24305	130.897428	172.16.2.2	172.16.2.4	TCP	1514 ✓	80 → 50839 [ACK] Seq=3381 Ack=822 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
24306	130.897530	172.16.2.2	172.16.2.4	TCP	1514 ✓	80 → 50839 [PSH, ACK] Seq=4841 Ack=822 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
24307	130.897547	172.16.2.4	172.16.2.2	TCP	54 ✓	50839 → 80 [ACK] Seq=822 Ack=6301 Win=2102272 Len=0
24308	130.897611	172.16.2.2	172.16.2.4	TCP	1514 ✓	80 → 50839 [ACK] Seq=6301 Ack=822 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
24309	130.897700	172.16.2.2	172.16.2.4	TCP	1514 ✓	80 → 50839 [PSH, ACK] Seq=7761 Ack=822 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
24310	130.897711	172.16.2.4	172.16.2.2	TCP	54 ✓	50839 → 80 [ACK] Seq=822 Ack=9221 Win=2102272 Len=0
24311	130.898066	172.16.2.2	172.16.2.4	HTTP	254 ✓	HTTP/1.1 200 OK (PNG)
24312	130.944472	172.16.2.4	172.16.2.2	TCP	54 ✓	50839 → 80 [ACK] Seq=822 Ack=9421 Win=2102016 Len=0
24313	131.565613	172.16.2.4	172.16.2.1	DNS	75 ✓	Standard query 0x7eb7 A bugs.debian.org
24314	131.565922	172.16.2.4	172.16.2.1	DNS	76 ✓	Standard query 0x004b A httpd.apache.org
24315	131.600858	172.16.2.4	172.16.2.2	HTTP	429 ✓	GET /favicon.ico HTTP/1.1
24316	131.601819	172.16.2.2	172.16.2.4	TCP	60 ✓	80 → 50839 [ACK] Seq=9421 Ack=1197 Win=64128 Len=0
24317	131.602237	172.16.2.2	172.16.2.4	HTTP	542 ✓	HTTP/1.1 404 Not Found (text/html)
24318	131.629120	172.16.2.1	172.16.2.4	DNS	92 ✓	Standard query response 0x004b A httpd.apache.org A 151.101.2.132

Figure 5.1 – Threat Detected



5.2 – Suricata Alerts



Works Cited

"Broadcast Address." Edited by Michael Gregg, *Broadcast Address - an Overview | ScienceDirect Topics*, 2006, https://www.sciencedirect.com/topics/computer-science/broadcast-address.

Packet data stripped by "TCP Invalid Checksum" IPS protection. Packet data stripped by "TCP invalid checksum" IPS protection. (n.d.). Retrieved October 9, 2022, from https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk173191#:~:text=The%20TCP%20Invalid%20Checksum%20protection,ACK%20to%20preserve%20the%20stream.

Seaton, Jennifer, and Margaret Rouse. "What Is Port 80? - Definition from Techopedia." *Techopedia.com*, 11 Mar. 2022, https://www.techopedia.com/definition/15709/port-80.