Project:
Network Administration
Network Scan, Information Collection, and Report

By Evelyn Wolfe

Table of Contents

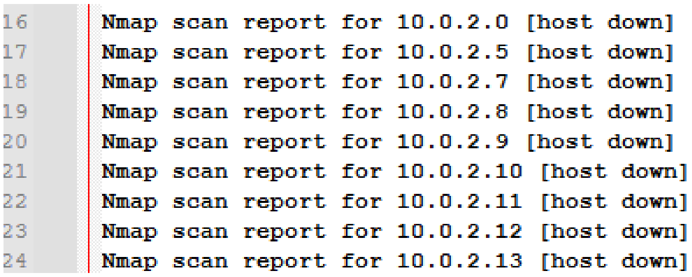
Introduction of Network Administration Summary:

With the knowledge of the past weeks put into play, this is my example of how the executive summary would reflect on network administrative level. This is solely reflective of my understanding of the network, and how tools such as Wireshark, Zenmap, and nmap play an effect of how a report would be generated and reviewed.

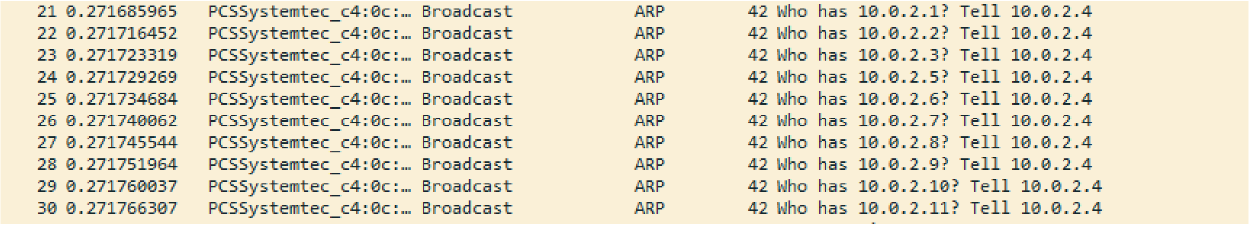
Within this report, the discovery of the following Virtual Machines, Windows, Linux Ubuntu are present. However, Kali is present by only IP of 10.0.2.4, this is due to the limitation of the ports and lack thereof open ports. The server has no ports open, and thus the details are missing. I consulted the mentors regarding this, and received confirmation that without ports modifications, the report would show limitations.

The details of the Linux Ubuntu and Windows 11 VM are detailing the IP, MAC Addresses, open ports. Further are the detailing reports of Zenmap, and Wireshark examples are included, but further dive into these reports through the share: Zenmap Scan and Wireshark Scan. These reports will be referenced numerous times throughout the report, and easy links will be cited. I began the information collection on the Kali machine – 10.0.2., on the network 10.0.2.0/24(Local Host of 10.0.2.1), with the Kali machine being the point of origin.

The following screenshots corresponded with the test that was conducted in Zenmap and reflected with in Wireshark as far as communication from one server to another.



```
Nmap scan report for 10.0.2.0 [host down]
Nmap scan report for 10.0.2.5 [host down]
Nmap scan report for 10.0.2.7 [host down]
Nmap scan report for 10.0.2.8 [host down]
Nmap scan report for 10.0.2.9 [host down]
Nmap scan report for 10.0.2.10 [host down]
Nmap scan report for 10.0.2.11 [host down]
Nmap scan report for 10.0.2.12 [host down]
Nmap scan report for 10.0.2.13 [host down]
```



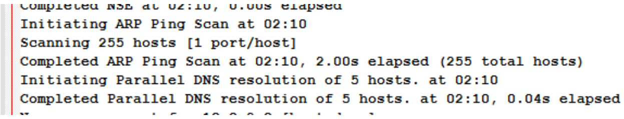
```
21 0.271685965   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.1? Tell 10.0.2.4
22 0.271716452   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.2? Tell 10.0.2.4
23 0.271723319   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.3? Tell 10.0.2.4
24 0.271729269   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.5? Tell 10.0.2.4
25 0.271734684   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.6? Tell 10.0.2.4
26 0.271740062   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.7? Tell 10.0.2.4
27 0.271745544   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.8? Tell 10.0.2.4
28 0.271751964   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.9? Tell 10.0.2.4
29 0.271760037   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.10? Tell 10.0.2.4
30 0.271766307   PCSSystemtec_c4:0c:… Broadcast          ARP        42 Who has 10.0.2.11? Tell 10.0.2.4
```

Network Scan and Collection of Information:

Utilization of Zenmap, and nmap, I was able to collect the following details regarding the network I have configured for my Virtual Machines – 10.0.2.0/24, while running Wireshark. The commands for Zenmap can be found

Please be advised the scan was conducted on Kali VM, and there are no ports open on this VM, causing no detailed information to be included in this report regarding the VM/Server.

The ARP Ping Scan elapse time was 2.00 seconds, with 255 total host.



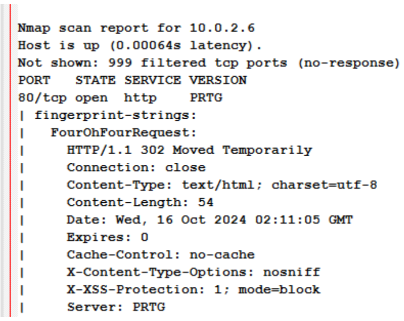
```
Completed NSE at 02:10, 0.00s elapsed
Initiating ARP Ping Scan at 02:10
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 02:10, 2.00s elapsed (255 total hosts)
Initiating Parallel DNS resolution of 5 hosts. at 02:10
Completed Parallel DNS resolution of 5 hosts. at 02:10, 0.04s elapsed
```

Windows VM - IP: 10.0.2.6

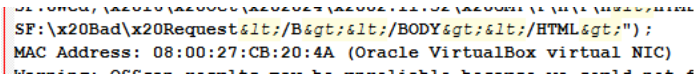
Version: PRTG

MAC: 08:00:27:CB:20:4A

Open Ports: 80/tcp



```
Nmap scan report for 10.0.2.6
Host is up (0.00064s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
80/tcp open  http    PRTG
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 302 Moved Temporarily
|     Connection: close
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 54
|     Date: Wed, 16 Oct 2024 02:11:05 GMT
|     Expires: 0
|     Cache-Control: no-cache
|     X-Content-Type-Options: nosniff
|     X-XSS-Protection: 1; mode=block
|     Server: PRTG
```

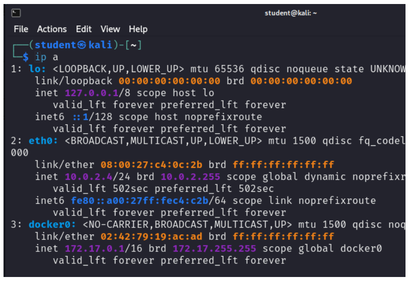


```
SF:\x20Bad\x20Request&lt;/B&gt;&lt;/BODY&gt;&lt;/HTML&gt;");
MAC Address: 08:00:27:CB:20:4A (Oracle VirtualBox virtual NIC)
```

Linux Kali – IP:10.0.2.4

MAC: 08:00:27:C4:0C:2B

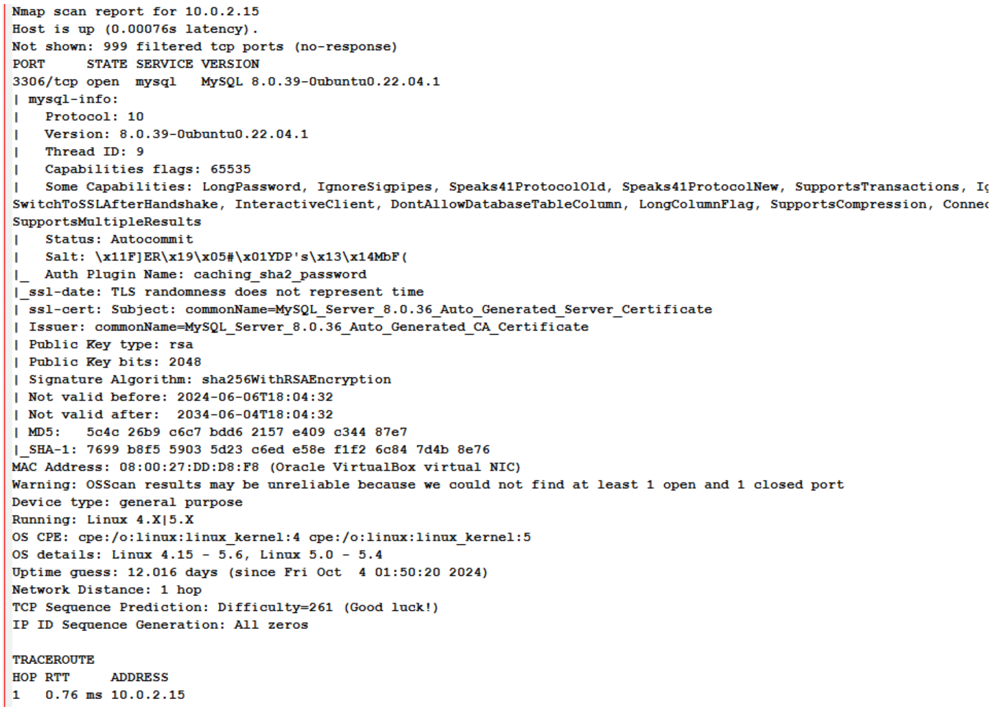
Open Ports: None



```
┌──(student㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
000
    link/ether 08:00:27:c4:0c:2b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic noprefixr
       valid_lft 502sec preferred_lft 502sec
    inet6 fe80::a00:27ff:fec4:c2b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noq
    link/ether 02:42:79:19:ac:ad brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

Linux Ubuntu – IP:10.0.2.15

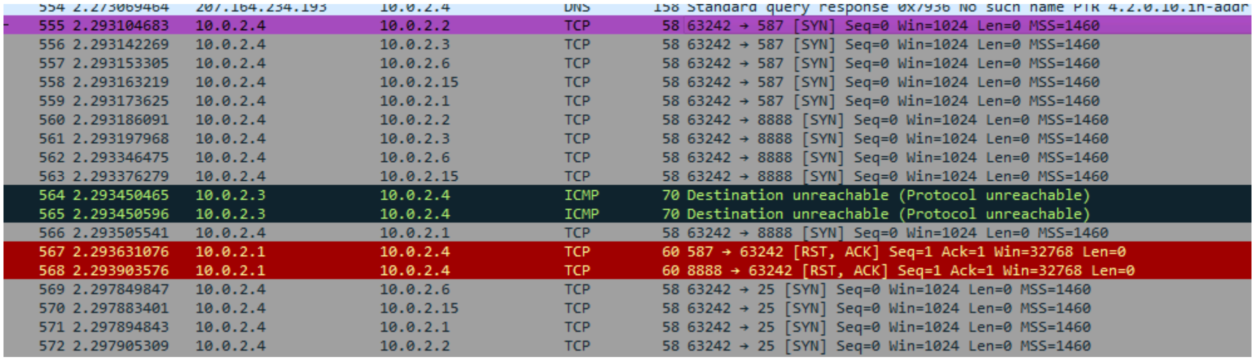
Version: 8.0.37-0ubuntu0.22.04.3
MAC: 08:00:27:DD:D8:F8
Open Ports: 3306/tcp



```
Nmap scan report for 10.0.2.15
Host is up (0.00076s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
3306/tcp open  mysql   MySQL 8.0.39-0ubuntu0.22.04.1
| mysql-info:
|   Protocol: 10
|   Version: 8.0.39-0ubuntu0.22.04.1
|   Thread ID: 9
|   Capabilities flags: 65535
|   Some Capabilities: LongPassword, IgnoreSigpipes, Speaks41ProtocolOld, Speaks41ProtocolNew, SupportsTransactions, Ig
SwitchToSSLAfterHandshake, InteractiveClient, DontAllowDatabaseTableColumn, LongColumnFlag, SupportsCompression, Connec
SupportsMultipleResults
|   Status: Autocommit
|   Salt: \x11F]ER\x19\x05#\x01YDP's\x13\x14MbF(
|_  Auth Plugin Name: caching_sha2_password
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=MySQL_Server_8.0.36_Auto_Generated_Server_Certificate
| Issuer: commonName=MySQL_Server_8.0.36_Auto_Generated_CA_Certificate
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2024-06-06T18:04:32
| Not valid after:  2034-06-04T18:04:32
| MD5:   5c4c 26b9 c6c7 bdd6 2157 e409 c344 87e7
|_SHA-1: 7699 b8f5 5903 5d23 c6ed e58e f1f2 6c84 7d4b 8e76
MAC Address: 08:00:27:DD:D8:F8 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6, Linux 5.0 - 5.4
Uptime guess: 12.016 days (since Fri Oct  4 01:50:20 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros

TRACEROUTE
HOP RTT     ADDRESS
1   0.76 ms 10.0.2.15
```

Here is the Wireshark showing communication between the different VM Listed. Including the indication of the TCP Session is initiated with this communication.



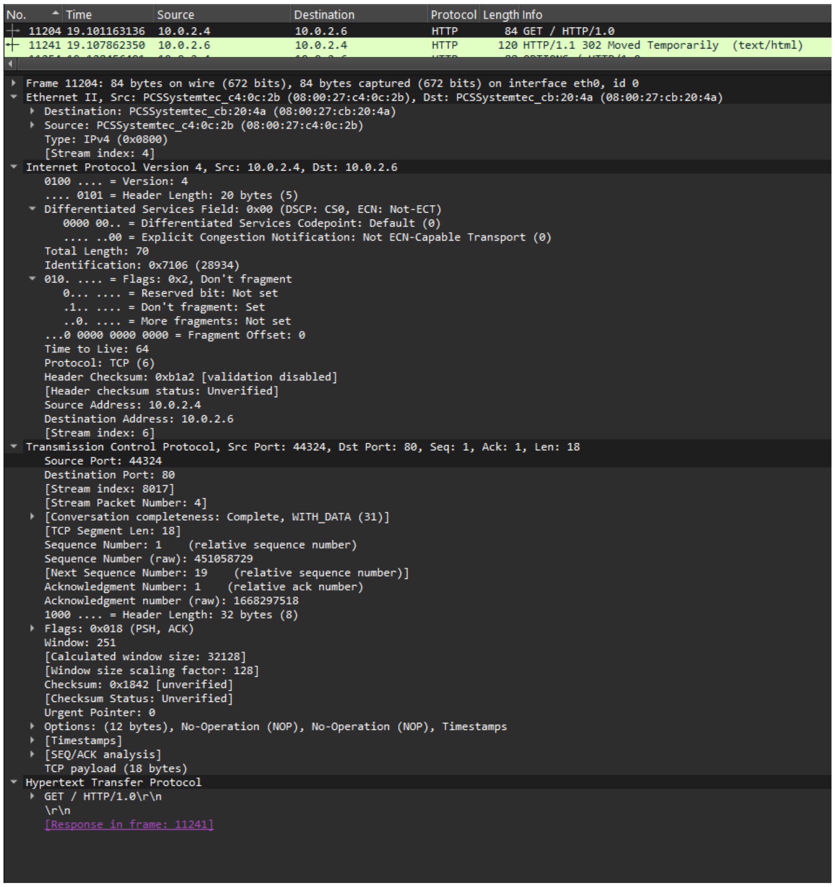
```
    554 2.273069464   207.164.234.193   10.0.2.4    DNS    158 Standard query response 0x7936 No such name PTR 4.2.0.10.in-addr
-   555 2.293104683   10.0.2.4          10.0.2.2    TCP     58 63242 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    556 2.293142269   10.0.2.4          10.0.2.3    TCP     58 63242 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    557 2.293153305   10.0.2.4          10.0.2.6    TCP     58 63242 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    558 2.293163219   10.0.2.4          10.0.2.15   TCP     58 63242 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    559 2.293173625   10.0.2.4          10.0.2.1    TCP     58 63242 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    560 2.293186091   10.0.2.4          10.0.2.2    TCP     58 63242 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    561 2.293197968   10.0.2.4          10.0.2.3    TCP     58 63242 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    562 2.293346475   10.0.2.4          10.0.2.6    TCP     58 63242 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    563 2.293376279   10.0.2.4          10.0.2.15   TCP     58 63242 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    564 2.293450465   10.0.2.3          10.0.2.4    ICMP    70 Destination unreachable (Protocol unreachable)
    565 2.293450596   10.0.2.3          10.0.2.4    ICMP    70 Destination unreachable (Protocol unreachable)
    566 2.293505541   10.0.2.4          10.0.2.1    TCP     58 63242 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    567 2.293631076   10.0.2.1          10.0.2.4    TCP     60 587 → 63242 [RST, ACK] Seq=1 Ack=1 Win=32768 Len=0
    568 2.293903576   10.0.2.1          10.0.2.4    TCP     60 8888 → 63242 [RST, ACK] Seq=1 Ack=1 Win=32768 Len=0
    569 2.297849847   10.0.2.4          10.0.2.6    TCP     58 63242 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    570 2.297883401   10.0.2.4          10.0.2.15   TCP     58 63242 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    571 2.297894843   10.0.2.4          10.0.2.1    TCP     58 63242 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    572 2.297905309   10.0.2.4          10.0.2.2    TCP     58 63242 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

Looking at the initiation of line 555, this is a TCP Protocol with SYN, which is the standard WireShark uses to indicate the TCP session has begun.

OSI Model:

Here is an amazing example of the Kali machine initiating contact with the other machines within the VM under the NAT Network. This demonstrates the initiation of the handshake, and the bridge of the TCP standards under the Transport protocol.

I really feel that this example of line 11204 is a great overview of the OSI Model in play from start to finish. This is a communication between Kali 10.0.2.4 to Windows 10.0.2.6.



```
No.      Time          Source         Destination    Protocol Length Info
11204 19.101163136   10.0.2.4       10.0.2.6       HTTP       84 GET / HTTP/1.0
11241 19.107862350   10.0.2.6       10.0.2.4       HTTP      120 HTTP/1.1 302 Moved Temporarily  (text/html)

Frame 11204: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_c4:0c:2b (08:00:27:c4:0c:2b), Dst: PCSSystemtec_cb:20:4a (08:00:27:cb:20:4a)
   Destination: PCSSystemtec_cb:20:4a (08:00:27:cb:20:4a)
   Source: PCSSystemtec_c4:0c:2b (08:00:27:c4:0c:2b)
   Type: IPv4 (0x0800)
   [Stream index: 4]
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.6
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
   Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
   Total Length: 70
   Identification: 0x7106 (28934)
   010. .... = Flags: 0x2, Don't fragment
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
   ...0 0000 0000 0000 = Fragment Offset: 0
   Time to Live: 64
   Protocol: TCP (6)
   Header Checksum: 0xb1a2 [validation disabled]
   [Header checksum status: Unverified]
   Source Address: 10.0.2.4
   Destination Address: 10.0.2.6
   [Stream index: 6]
Transmission Control Protocol, Src Port: 44324, Dst Port: 80, Seq: 1, Ack: 1, Len: 18
   Source Port: 44324
   Destination Port: 80
   [Stream index: 8017]
   [Stream Packet Number: 4]
   [Conversation completeness: Complete, WITH_DATA (31)]
   [TCP Segment Len: 18]
   Sequence Number: 1    (relative sequence number)
   Sequence Number (raw): 451058729
   [Next Sequence Number: 19    (relative sequence number)]
   Acknowledgment Number: 1    (relative ack number)
   Acknowledgment number (raw): 1668297518
   1000 .... = Header Length: 32 bytes (8)
   Flags: 0x018 (PSH, ACK)
   Window: 251
   [Calculated window size: 32128]
   [Window size scaling factor: 128]
   Checksum: 0x1842 [unverified]
   [Checksum Status: Unverified]
   Urgent Pointer: 0
   Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
   [Timestamps]
   [SEQ/ACK analysis]
   TCP payload (18 bytes)
Hypertext Transfer Protocol
   GET / HTTP/1.0\r\n
   \r\n
   [Response in frame: 11241]
```

The Example

This example relays the Application layer in the HTTP (Hypertext Transfer Protocol,) This takes the PDU Packet “Data” into “Frame” Via the Transport Layer, Network Layer, and Data Link Layer.

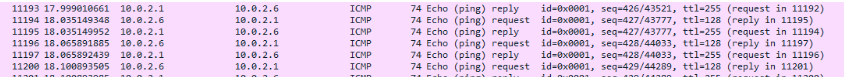
These layers are identified as the following:

- Application Layer via HTTP
    - This is identified by the Hypertext Transfer Protocol
- Transport Layer is the Transmission Control Protocol

    - This is identified by the Ports and the Segments (Seg: 1)
    - This section adds a header to the packet “Layer 4 Header”
- Network Layer is the Internet Protocol Version 4, 10.0.2.4, Dst: 10.0.2.6
    - This is identifying by adding an address to the segments to ensure their delivery
    - This section ads a header to the packet “Layer 3 Header”
- Data Link Layer with Ethernet II, Src PCSSystemec_c4:0c:2b
    - These are the Physical Addresses – MAC Address, IP Addresses.
    - This section adds a header and tailer to the packet for identification. “Layer 2 Header” “Layer 2 Tailer”

The packet will look like this based on the OSI Model “Layer 2 Header” “Layer 3 Header” “Layer 4 Header” “User Data” “Layer 2 Tailer”

Furthermore, the ICMP is also heavily active later in the recording indicating more transfer requests and replies. This further proof of the OSI Model communication.

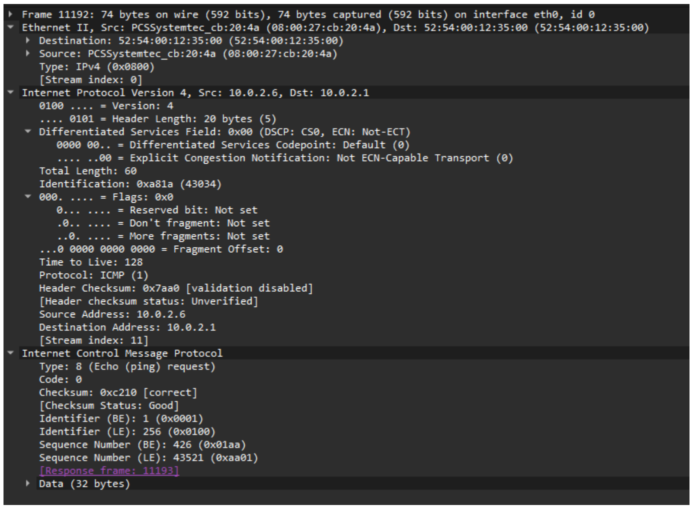


```
11193 17.999010661  10.0.2.1            10.0.2.6            ICMP       74 Echo (ping) reply    id=0x0001, seq=426/43521, ttl=255 (request in 11192)
11194 18.035149348  10.0.2.6            10.0.2.1            ICMP       74 Echo (ping) request  id=0x0001, seq=427/43777, ttl=128 (reply in 11195)
11195 18.035149952  10.0.2.1            10.0.2.6            ICMP       74 Echo (ping) reply    id=0x0001, seq=427/43777, ttl=255 (request in 11194)
11196 18.065891885  10.0.2.6            10.0.2.1            ICMP       74 Echo (ping) request  id=0x0001, seq=428/44033, ttl=128 (reply in 11197)
11197 18.065892439  10.0.2.1            10.0.2.6            ICMP       74 Echo (ping) reply    id=0x0001, seq=428/44033, ttl=255 (request in 11196)
11200 18.100893505  10.0.2.6            10.0.2.1            ICMP       74 Echo (ping) request  id=0x0001, seq=429/44289, ttl=128 (reply in 11201)
```

Let’s look further into Line 11192.

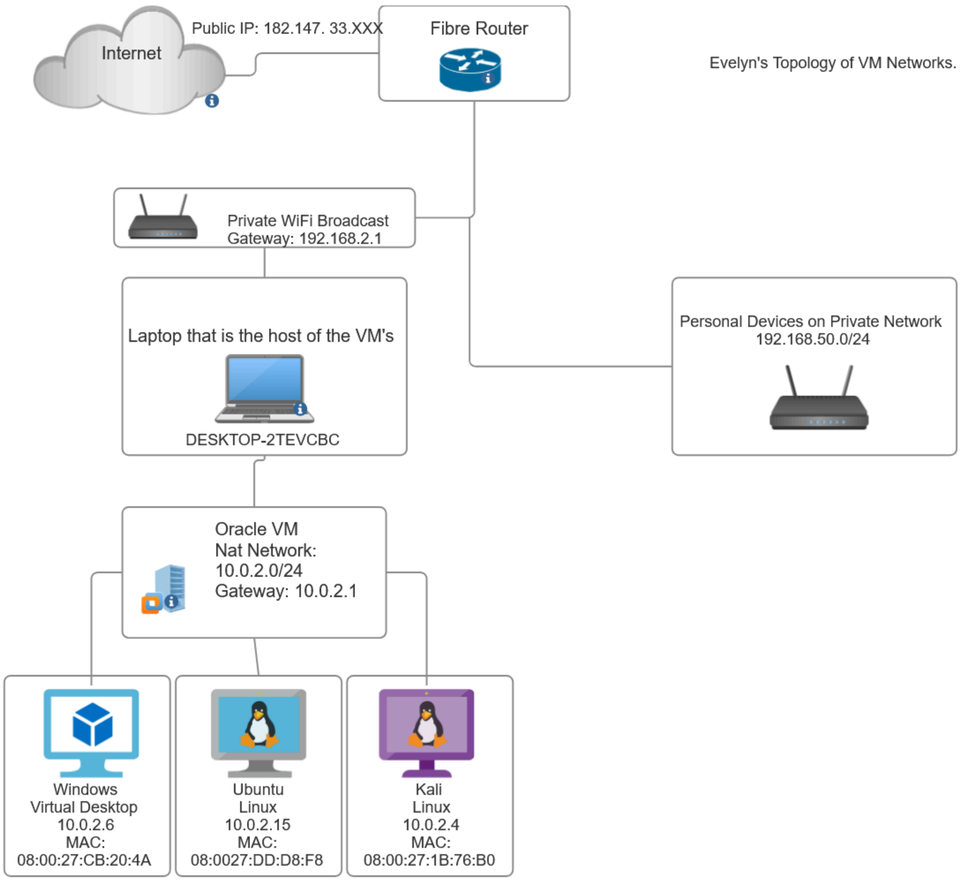
Under the Ethernet Layer, one can clearly see the transmission of Data Link Layer, the Ethernet is the physical address of both the source, and destination, IP and MAC addresses. This would give the packet a Layer 2 Header, and Tailer

Both the Internet Protocol Version 4, and Internet Control Message Protocol would fall under the network layer, give the packet a Layer 3 Header.



```
Frame 11192: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_cb:20:4a (08:00:27:cb:20:4a), Dst: 52:54:00:12:35:00 (52:54:00:12:35:00)
   Destination: 52:54:00:12:35:00 (52:54:00:12:35:00)
   Source: PCSSystemtec_cb:20:4a (08:00:27:cb:20:4a)
   Type: IPv4 (0x0800)
   [Stream index: 0]
Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.1
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
   Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
   Total Length: 60
   Identification: 0xa81a (43034)
   000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
   ...0 0000 0000 0000 = Fragment Offset: 0
   Time to Live: 128
   Protocol: ICMP (1)
   Header Checksum: 0x7aa0 [validation disabled]
   [Header checksum status: Unverified]
   Source Address: 10.0.2.6
   Destination Address: 10.0.2.1
   [Stream index: 11]
Internet Control Message Protocol
   Type: 8 (Echo (ping) request)
   Code: 0
   Checksum: 0xc210 [correct]
   [Checksum Status: Good]
   Identifier (BE): 1 (0x0001)
   Identifier (LE): 256 (0x0100)
   Sequence Number (BE): 426 (0x01aa)
   Sequence Number (LE): 43521 (0xaa01)
   [Response frame: 11193]
   Data (32 bytes)
```

Example of Topology of Evelyn's Network – VM Related



In this display, this shows how the network segmentation is in effect against the other devices on my personal network.

The network enters the household and is split from the router into two different fields, the WiFi that the laptop uses, and the private network the other devices use in the household. Diving even further into the VMs, the network for the three VM are joined by a NAT Network within Oracle. This isolates the Virtual Machines into the 10.0.2.0/24, keeping everything separated.

The only change to the network I would make at this point is a physical firewall for an extra level of security.

References and Citation

Zenmap and NMAP Citation/Reference

Lyon, G (2009, July) *Nmap Network Scanning*
Retrieved from https://nmap.org/book/zenmap-scanning.html

Wireshark Citation/Reference

Wireshark Cheat Sheet (2019, June)
Retrieved from
https://cdn.comparitech.com/wp-content/uploads/2019/06/Wireshark-Cheat-Sheet.pdf

OSI Model Citation/Reference

Postel, J., (1981, September) *Transmission Protocol Data Internet Program Protocol Specificaiton*
Information Science Institute – University of Southern California
Retrieved from https://www.rfc-editor.org/rfc/rfc793#section-3.4

Hoang, A. (2015, March) *Viewing OSI Layers on Wireshark*
Retreived from
https://medium.com/the-cabin-coder/viewing-osi-layers-on-wireshark-a51b77cfbd72

Reference to Citation/Reference

Citing a Website without Authors (2022, February)
Cite This For Me
Retrieved from
https://www.citethisforme.com/citation-generator/citation-basics/citing-website-without-author