Analysis of Network Packets and Security Enhancements

Stanley Shaw

October 30, 2024

Abstract

This report presents a comprehensive analysis of a provided PCAP file, with the aim of discovering the network's architecture and operational dynamics. Additionally, the report proposes a redesign of the network to establish zones of trust, enhance security measures, and configure zone-policy firewalls, substantiated by findings from the packet analysis.

Contents

1 Introduction

Packet capture files can be used in order to map the topology of networks, this can allow cyber security professionals to gain an insight into the operating, performance and security of a network. By examining the networks packets I will gain insight into what improvements can be made in order to maintain the networks ongoing security and optimisation.

2 Network Discovery

2.1 Data Collection Period

Firstly, in order find the date and time at which the packets where sent I uploaded the packet capture file into the Wireshark software. This allowed me too inspect the first and the last packet sent and as a result I concurred that all packets had been captured between 21.10.2015 and 22.10.2015. Furthermore, the time they were captured ranged between 11:10 pm and 2:17 pm the next day this can be seen below:

```
Frame 1: 153 bytes on wire (1224 bits), 153 bytes captured (1224 bits)
   Encapsulation type: Ethernet (1)
   Arrival Time: Oct 21, 2015 23:10:34.995270000 GMT Summer Time
```

Figure 1: First frame time stamp

```
Frame 2274742: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
   Encapsulation type: Ethernet (1)
   Arrival Time: Oct 22, 2015 14:17:36.828919000 GMT Summer Time
```

Figure 2: Last frame time stamp

2.2 IPV4 Addresses

I then moved onto finding all IPV4 addresses of host devices on the network. Moreover, I separated them into public and private IPV4 addresses allowing me to figure out the devices within the network. I also sorted all active devices into subnets and calculated there subnet mask the results can be seen below:

Public IPV4 Addresses	
8.8.8.8	17.110.224.213
17.110.230.30	17.130.137.73
17.130.137.75	17.253.34.253
17.253.54.251	21.2.2.2
52.4.151.114	52.5.95.205
54.210.217.83	54.241.179.26
74.125.205.188	77.245.33.76
83.140.27.11	93.158.94.210
93.158.110.200	93.158.110.218
108.160.163.110	141.82.217.52
173.252.90.4	192.195.142.13
192.195.142.14	193.182.190.178
193.209.237.4	199.16.156.48
199.16.156.70	199.16.156.72
199.16.156.198	199.16.156.231

Table 1: List of Public IPV4 Addresses

Private IPV4 Addresses			
10.10.10.10	10.10.10.20	10.10.10.30	10.100.152.10
10.100.152.15	10.100.152.119	10.100.152.128	10.100.158.168
10.100.158.185	10.100.159.27	10.100.159.125	10.100.159.151
10.100.159.207	10.100.159.218	10.100.159.227	10.100.159.228
10.100.159.247	10.100.159.253	10.218.104.244	172.16.184.40
192.168.0.3	192.168.1.2	192.168.1.10	192.168.1.68
192.168.1.71	192.168.1.79	192.168.2.21	192.168.2.22
192.168.2.44	192.168.2.53	192.168.2.64	192.168.2.110
192.168.2.133	192.168.2.137	192.168.2.166	192.168.2.199
192.168.57.2	192.168.57.3	192.168.88.1	192.168.88.2
192.168.88.15	192.168.88.20	192.168.88.25	192.168.88.30
192.168.88.49	192.168.88.50	192.168.88.51	192.168.88.52
192.168.88.53	192.168.88.54	192.168.88.55	192.168.88.60
192.168.88.61	192.168.88.75	192.168.88.80	192.168.88.85
192.168.88.95	192.168.88.100	192.168.88.105	192.168.88.115
192.168.88.130	192.168.88.254	192.168.89.1	192.168.89.2
192.168.143.1	192.168.143.155	192.168.143.254	

Table 2: List of Private IPV4 Addresses

Subnet	**Subnet Mask**	**Broadcast Address**
10.10.10.0/27	255.255.255.224	10.10.10.31
10.100.152.0/24	255.255.255.0	10.100.152.255
192.168.1.10/30	255.255.255.240	192.168.1.12
192.168.2.0/24	255.255.255.0	192.168.2.255
192.168.88.0/23	255.255.255.0	192.168.88.255

Table 3: All active subnets in the network

2.3 Networking Devices and VLANs

After discovering all endpoints on the network I then moved onto looking for networking devices and VLANs within the network. Firstly, I searched for routing protocol packets as this would allow me too easily identify routers within the network however, no routing packets were displayed in Wireshark this can be seen below:

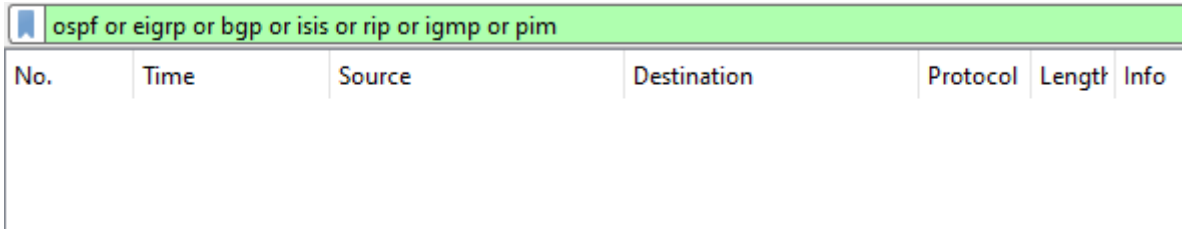

Figure 3: Routing packets in Wireshark

This suggests that static routing is used instead as this doesn't generate routing traffic like dynamic routing protocols do. In order to find networking devices I instead have to inspect the traffic flows to and from individual endpoints. Routing devices are devices that route packets between different networks they can be differentiated from traditional endpoints by analyzing the conversations that take place between the networks this can be seen below:

Address A	Address B	Packets	Bytes	Total Packets	Percent Filtered
17.110.230.30	10.100.158.185	8	3 kB	8	100.00%

Figure 4: Packets routed between different Networks in Wireshark

While this does show me routing, I had to filter the captured packets and look for ones that were transmitted between different networks this allowed me to locate the MAC address of the router and its corresponding IPV4 address. This can be seen below:

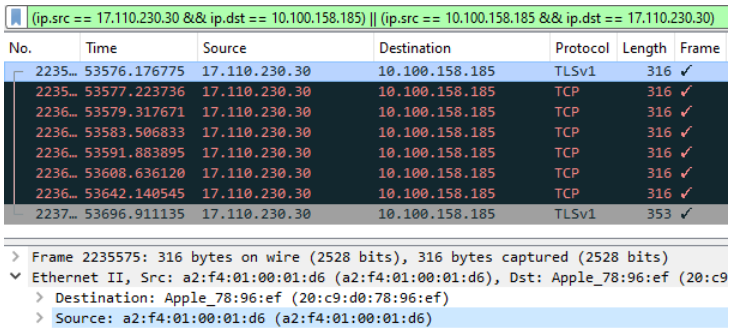

Figure 5: MAC address of a router

The source MAC address is the MAC address of the router that sits between the subnet 10.100.152.0/24 and the external internet (shown by the public nature of the IPV4 source), it has a corresponding IPV4 address of 10.100.152.1. This can be seen below:

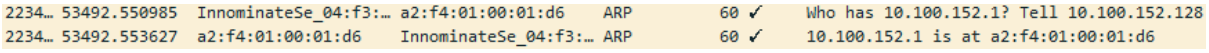

Figure 6: ARP response from router confirming IPV4 address of 10.100.152.1

In order to find the second router I looked through another conversation between different networks and I came across two more devices that were communicating between subnets 192.168.2.0/24 and 192.168.88.0/23 these can be seen below:

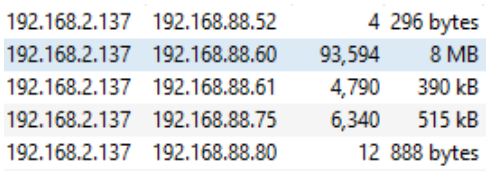

Figure 7: Packets travelling between different networks

After filtering the captured packets for ones moving between the two subnets I then moved onto inspecting the source address of the packets. This highlighted the mac address of the router as 00:07:7c:1a:61:83 as seen below:

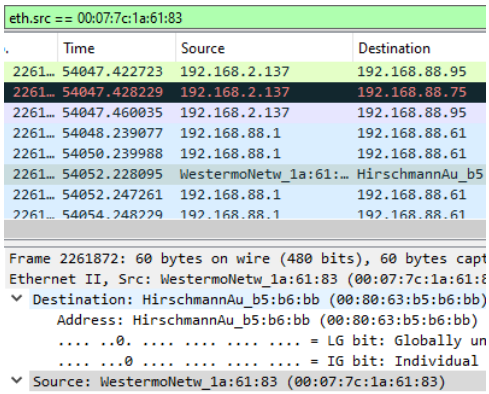

Figure 8: Packets travelling between subnets going through the MAC address 00:07:7c:1a:61:83

In order to find the IPV4 address of the router using the mac address I then moved onto inspecting the ARP packets from this mac address. This allowed me to locate an ARP packet which displayed the associated IPV4 address of the router as being 192.168.88.1.

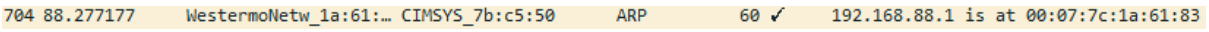

Figure 9: ARP response from router confirming IPV4 address of 192.168.88.1

In order to locate the third router within the network I used the same method of analysing traffic between networks and locating devices routing packets between them this can be seen below:

Address A	Address B	Packets	Bytes
10.10.10.30	172.16.184.40	8	6 kB

Figure 10: Conversation between subnet 10.10.10.0/27 and 192.168.1.10/32

After locating a conversation I then found the mac address of the intermediary device which was 04:18:d6:83:db:16.

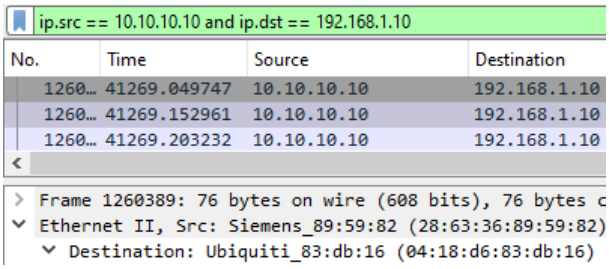

Figure 11: Mac address of the router

I then used ARP packets destined for this MAC address to find its associated IPV4 address of 10.10.10.1 this can be seen below:

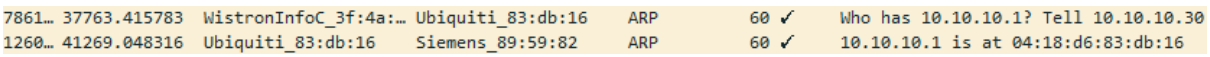

Figure 12: ARP response from router confirming IPV4 address of 10.10.10.1

After locating all routers within the network I then moved onto looking for VLAN's within the network this can be done through filtering the packets using the VLAN tag which displays any IEEE 802.1Q tagged packets. Using this method it was clear that there were no VLAN's were active within the network as no packets with this tag were displayed as seen below:

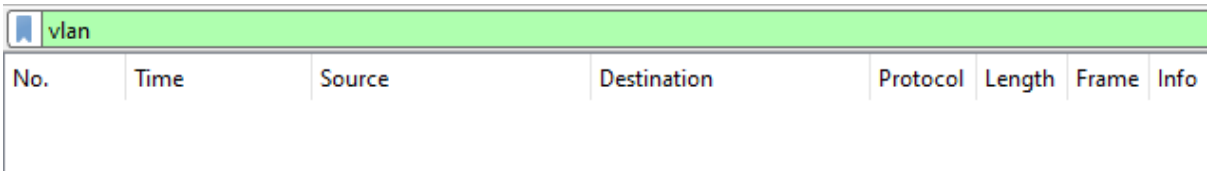

Figure 13: No packets displayed with the VLAN tag

2.4 Servers and Services

After identifying the Networking devices I then moved onto locating servers and services that were running within the network. I first used the Wireshark tool to locate a DHCP server at 10.100.152.10:

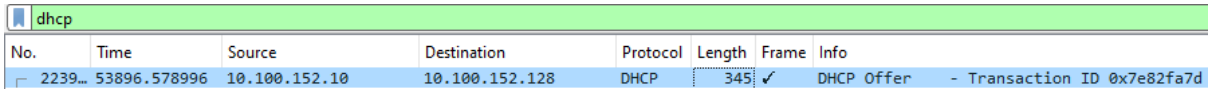

Figure 14: DHCP offer from 10.100.152.10

I then moved onto locating web servers on the network by using the WireShark filter http.response which locates all successful HTTP response packets sent from the web servers (Address B) to the client (Address A). By then using the conversations tab I was able to locate a number of web servers as seen below:

Figure 15: Filter for successful HTTP response packets

Address A	Address B	Packets	Bytes
192.168.2.137	192.168.88.20	10	1 kB
192.168.2.22	192.168.88.49	9	7 kB
192.168.2.199	192.168.88.51	15	18 kB
192.168.2.21	192.168.88.60	73	37 kB
192.168.2.199	192.168.88.61	2	2 kB
192.168.2.137	192.168.88.100	2	120 bytes
192.168.2.22	192.168.88.115	21	12 kB

Figure 16: All IPV4's serving HTTP responses

However, this does not include HTTPS response packets as these are encrypted in order to locate these devices I instead used a filter that looks for packets with a port source of 443 as seen below:

Figure 17: Filter for successful HTTPs response packets

Address A	Address B	Packets	Bytes
192.168.2.22	192.168.88.61	34	2 kB
192.168.2.22	192.168.88.75	58	4 kB
192.168.2.22	192.168.88.95	69	4 kB
192.168.2.22	192.168.88.115	44	3 kB
192.168.2.53	192.168.88.51	193	15 kB
192.168.2.53	192.168.88.60	77	6 kB

Figure 18: Filter for successful HTTPs response packets

After filtering traffic correctly I discovered all IPV4 addresses sending HTTP and HTTPS responses as expected there is a large amount of overlap between them. Below is the list of endpoints ordered by the protocols they use:

Both HTTP and HTTPS	**Only HTTP**	**Only HTTPS**
192.168.88.51	192.168.88.20	192.168.88.75
192.168.88.60	192.168.88.49	192.168.88.95
192.168.88.61	192.168.88.100	
192.168.88.115		

However, these devices may not all be functioning as web servers In order to find if they were primarily web servers I used the NetworkMiner tool to check if the devices had a web server banner as seen below:

Devices	**Contains web server banner**	**Web server software**
192.168.88.51	Yes	Microsoft-WinCE/5.0
192.168.88.60	Yes	GoAhead-Webs
192.168.88.61	Yes	GoAhead-Webs
192.168.88.115	Yes	GoAhead-Webs
192.168.88.95	Yes	GoAhead-Webs
192.168.88.20	Yes	NET+ARM Web Server/1.00
192.168.88.49	No	N/A
192.168.88.100	No	N/A
192.168.88.75	No	N/A

This suggests that only some of the devices serving http and https responses are serving as web servers and the others serve a different purpose. After completing this I then moved onto to looking at devices involved in ICS (Industrial control systems) which seem to play a prominent role in the system. This was done through the use of WireShark by filtering traffic for port 102 which plays a key role in monitoring and controlling ICS as seen below:

Address A	Address B	Packets	Bytes	Total Packets	Percent Filtered
192.168.1.10	10.10.10.10	96	7 kB	96	100.00%
10.10.10.30	10.10.10.10	43,135	3 MB	43,135	100.00%
10.10.10.20	10.10.10.10	326,505	34 MB	326,505	100.00%

Figure 19: Filter for traffic on port 102

This highlights that the main ICS seems to be located within the 10.10.10.0/27 subnet furthermore, on deeper inspection it seems that the device with the IPV4 10.10.10.10 is the primary ICS device while 10.10.10.30 is used to control the device and 10.10.10.20 is used to monitor the device this can be seen below:

ip.dst == 10.10.10.10 && ip.src == 10.10.10.20 && s7comm

No.	Time	Source	Destination	Protocol	Length	Frame	Info
1	0.000000	10.10.10.20	10.10.10.10	S7COMM	153	✓	ROSCTR:[Job] Function:[Read Var]
12	0.999852	10.10.10.20	10.10.10.10	S7COMM	153	✓	ROSCTR:[Job] Function:[Read Var]
20	1.999962	10.10.10.20	10.10.10.10	S7COMM	153	✓	ROSCTR:[Job] Function:[Read Var]

Figure 20: Traffic from 10.10.10.20 monitoring 10.10.10.10 using Read Var command

ip.dst == 10.10.10.10 && ip.src == 10.10.10.30 && s7comm

No.	Time	Source	Destination	Protocol	Length	Frame	Info
1262…	41509.003188	10.10.10.30	10.10.10.10	S7COMM	94	✓	ROSCTR:[Job] Function:[Write Var]
1262…	41524.186632	10.10.10.30	10.10.10.10	S7COMM	94	✓	ROSCTR:[Job] Function:[Write Var]
1276…	42824.883137	10.10.10.30	10.10.10.10	S7COMM	94	✓	ROSCTR:[Job] Function:[Write Var]

Figure 21: Traffic from 10.10.10.30 controlling 10.10.10.10 using Write Var command

After locating the ICS devices I then looked for FTP servers by filtering for FTP packets in WireShark this displayed two devices that were serving FTP responses 192.168.88.49 and 192.168.88.25. This can be seen below:

ftp

No.	Time	Source	Destination	Protocol	Length	Frame	Info
2752…	32911.978260	192.168.88.49	192.168.2.137	FTP	121	✓	Response:
2752…	32911.982107	192.168.88.49	192.168.2.137	FTP	80	✓	Response:

Figure 22: FTP response from 192.168.88.49

ftp

No.	Time	Source	Destination	Protocol	Length	Frame	Info
4054…	35107.935876	192.168.88.25	192.168.2.166	FTP	130	✓	Response:
4290…	35284.470859	192.168.88.25	192.168.2.166	FTP	130	✓	Response:

Figure 23: FTP response from 192.168.88.25

Additionally I identified two devices running SNMP (Simple Network Management Protocol) with IPV4's 192.168.2.22 (SNMP manager) and 192.168.88.30 (SNMP agent) as seen below:

udp.port == 161

No.	Time	Source	Destination	Protocol	Length	Frame	Info
1724…	46577.148644	192.168.2.22	192.168.88.30	SNMP	85	✓	get-next-request
1724…	46577.162572	192.168.88.30	192.168.2.22	SNMP	86	✓	get-response 1.3.

Figure 24: SNMP connection between 192.168.88.30 and 192.168.2.22

Finally, I identified a firewall operating between the subnets 192.168.88.0/23 and 192.168.2.0/24 with the IPV4 address 192.168.88.75 this was done through the use of the NetworkMiner tool which identified the device as a EAGLE20 Tofino which acts as a firewall. This can be seen below:

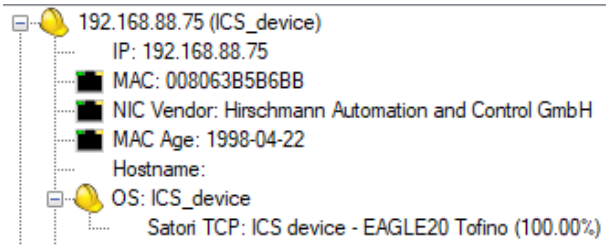

Figure 25: Network miner confirming the device is a EAGLE20 Tofino

2.5 Network Topology

After locating all services and servers on the network I then moved onto designing a network topology that matched the reconnaissance I have completed so far. This involved defining all the subnets with all the services I located within each subnet as seen below:

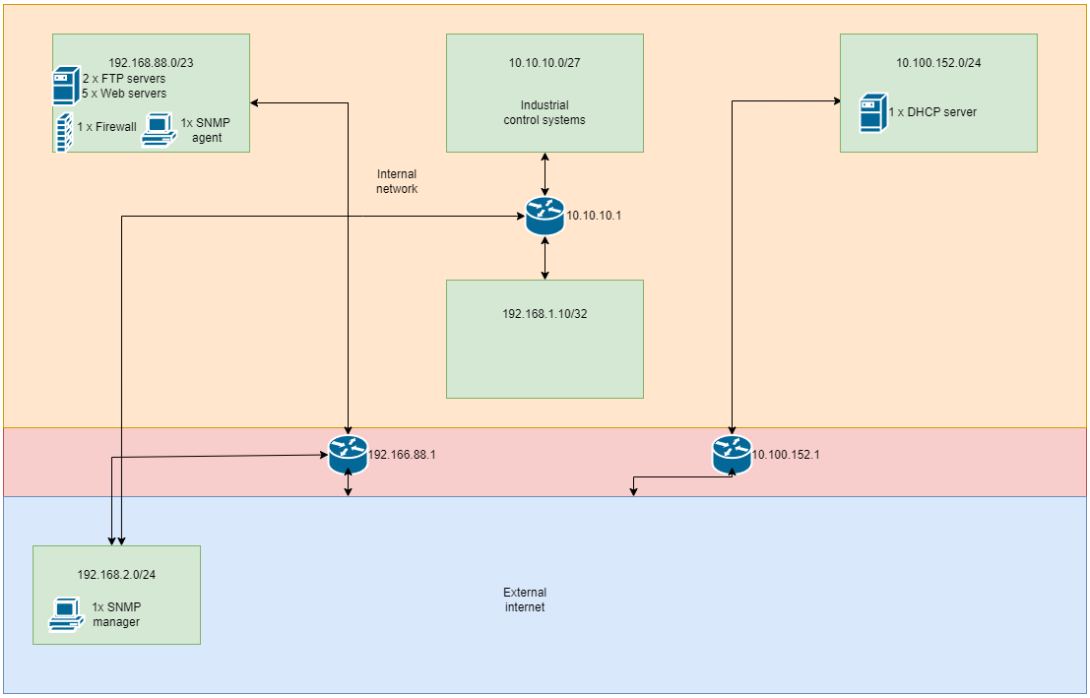

Figure 26: Subnet layout

This proposed layout is evidenced by the devices within the subnet 192.168.2.0/24 not having visible mac addresses. Furthermore, when analyzing the distance variable within NetworkMiner it is clear that packets destined for the 192.168.2.0/24 subnet have to complete a number of hops across the internet in order to reach there desired IP address. This suggests that the 192.168.2.0/24 subnet may be serving as a demilitarized zone that allows for connections from the wider internet. This is further evidenced by the OS on some devices in this network detecting NMAP scans. All evidence for my claims can be seen below:

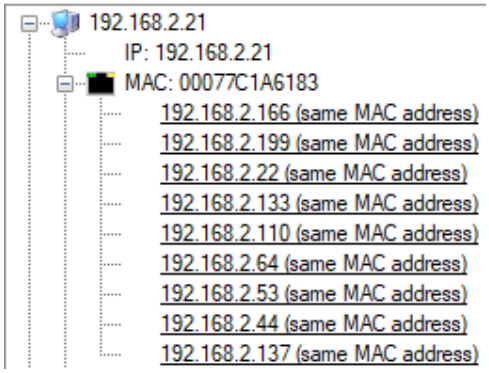

Figure 27: NetworkMiner cant detect mac addresses of devices in 192.168.2.0/24 subnet

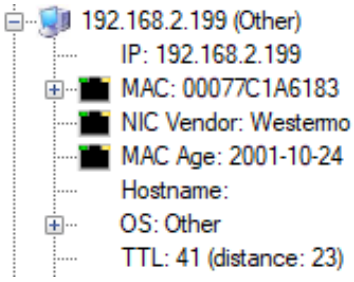

Figure 28: The high distance value of 23 suggests the network is externally located

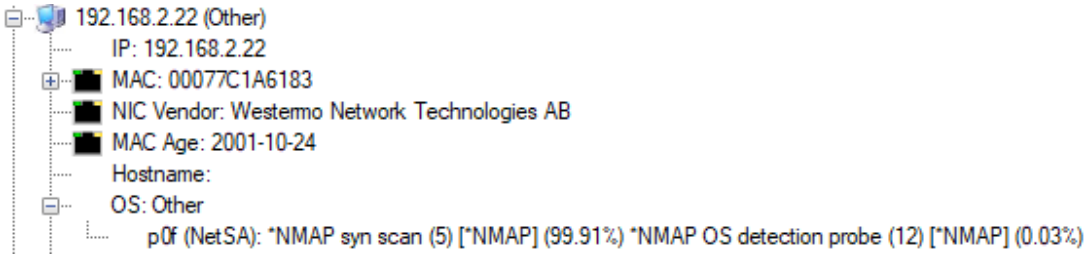

Figure 29: The device has detected a NMAP scan

2.6 Cisco Packet Tracer Simulation

After discovering and outlining the proposed network topology I created the network topology within Cisco packet tracer. I began placing all key devices including servers, routers, firewalls and a device for each subnet within the network this can be seen below:

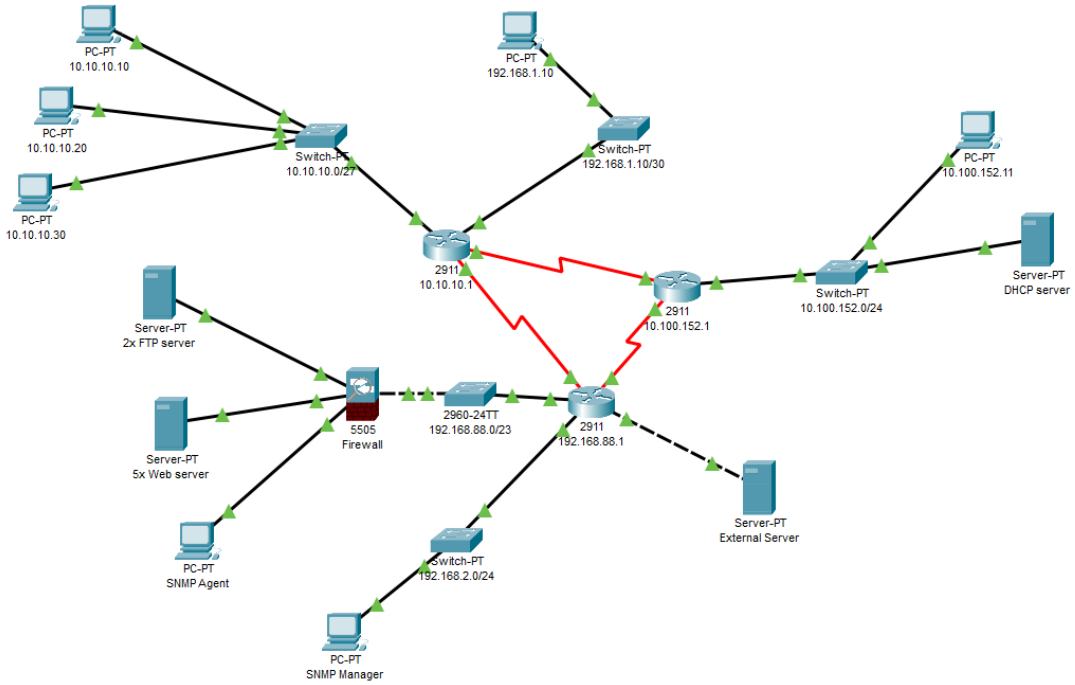

Figure 30: Network topology in Cisco packet tracer

I then moved onto assigning the correct IPV4's to all router interfaces within the network based of the captured packets. This can be seen below:

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface Gig0/0
Router(config-if)#ip address 192.168.88.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#end
```

Figure 31: Assigning IPV4 to 192.168.88.1 Gig0/0 Router interface

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface Gig0/0
Router(config-if)#ip address 10.100.152.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#end
```

Figure 32: Assigning IPV4 to 10.100.152.1 Gig0/0 Router interface

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface Gig0/0
Router(config-if)#ip address 10.10.10.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#end
```

Figure 33: Assigning IPV4 to 10.10.10.1 Gig0/0 Router interface

```
Router(config)#interface GigabitEthernet0/1
Router(config-if)#no ip address
Router(config-if)#ip address 192.168.1.1 255.255.255.0
```

Figure 34: Assigning IPV4 to 10.10.10.1 Gig0/1 Router interface

After correctly assigning IPV4's to each router interface I moved onto configuring the DHCP server in subnet 10.100.152.0/24 which assigns IPV4 addresses to every device within this subnet.

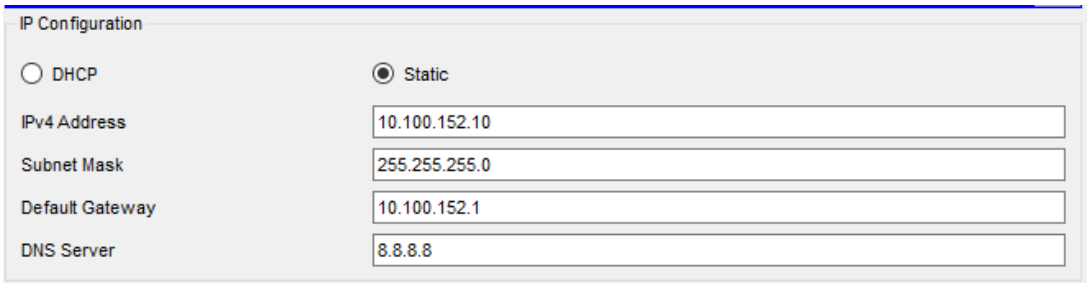

Figure 35: Configuring DHCP server IPV4 address

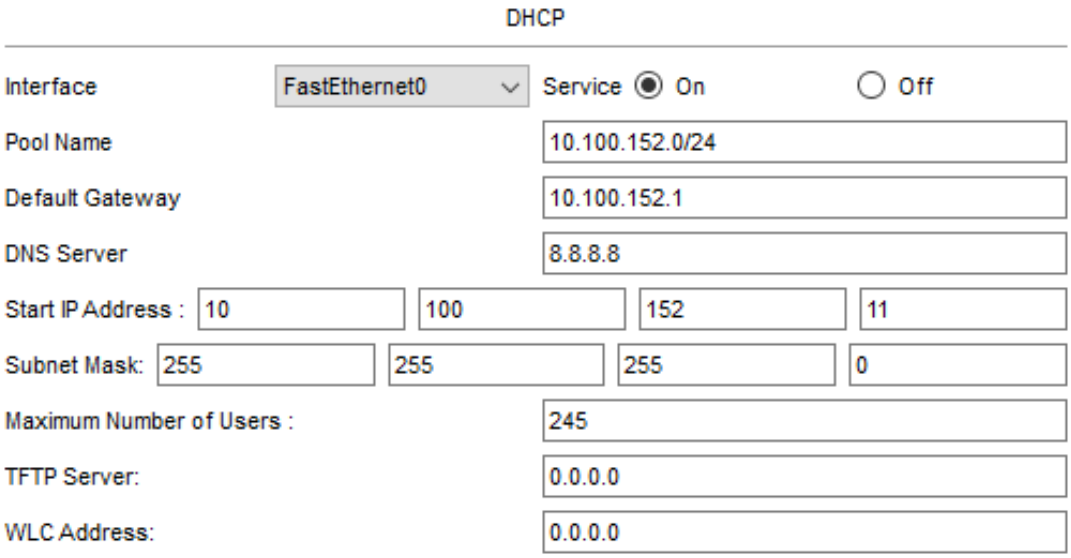

Figure 36: Configuring DHCP server pool

Figure 37: IPV4's successfully assigned to devices within the subnet

For devices located outside this subnet I instead manually assigned them the appropriate IPV4 address as seen below:

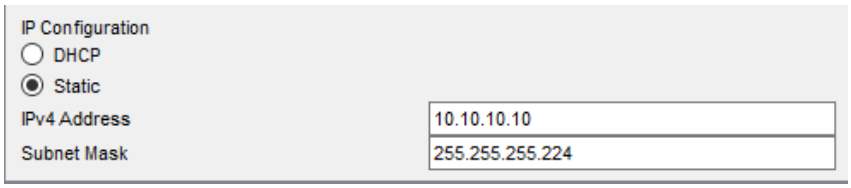

Figure 38: Assigning IP address to 10.10.10.10

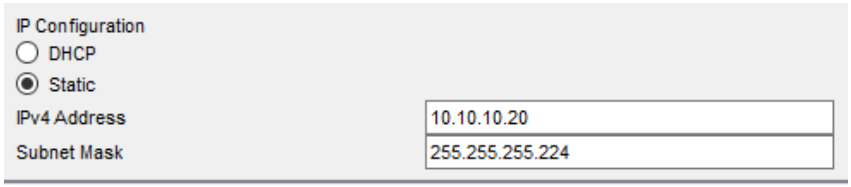

Figure 39: Assigning IP address to 10.10.10.20

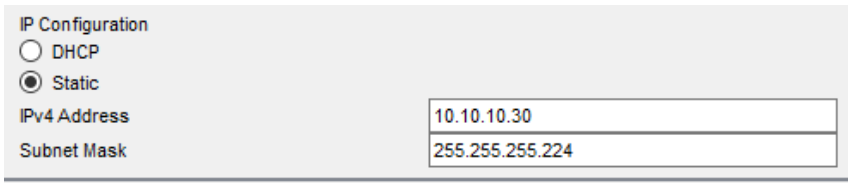

Figure 40: Assigning IP address to 10.10.10.30

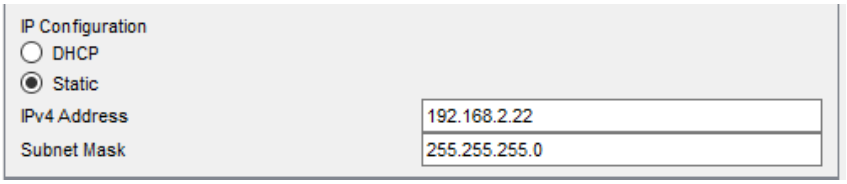

Figure 41: Assigning IP address to 192.168.2.22

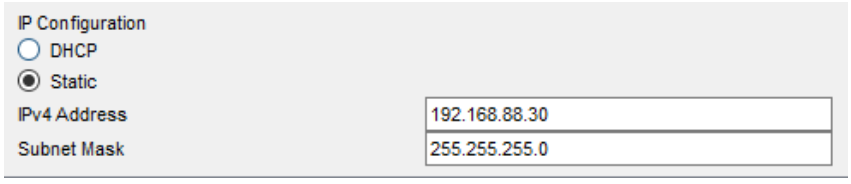

Figure 42: Assigning IP address to 192.168.88.30

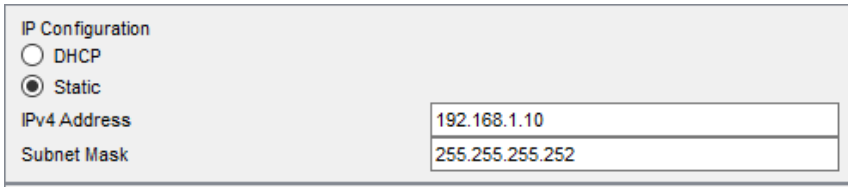

Figure 43: Assigning IP address to 192.168.1.10

Finally in order to complete the network topology I configured static routing between the different subnets this method was chosen as I could not find any OSPF or other routing protocol packets. This can be seen below:

```
Router>enable
Router#config terminal
Enter configuration commands, one per line.  End with CNTI
Router(config)#ip route 10.100.152.1 255.255.255.0 Gig0/0
%Default route without gateway, if not a point-to-point i
%Inconsistent address and mask
Router(config)#end
Router#write memory
```

Figure 44: Routing Configuration for 10.100.152.1

```
Router#enable
Router#config terminal
Enter configuration commands, one per line.  End with CNT
Router(config)#ip route 192.168.88.0 255.255.255.0 Gig0/0
%Default route without gateway, if not a point-to-point i
Router(config)#end
Router#write memory
```

Figure 45: Routing Configuration for 192.168.88.1

```
Router(config-if)#ip route 192.168.1.10 255.255.255.252 Gig0/2
%Default route without gateway, if not a point-to-point interfa
%Inconsistent address and mask
Router(config)#ip route 10.10.10.0 255.255.255.224 Gig0/0
```

Figure 46: Routing Configuration for 192.168.88.1

I then configured the serial interfaces and the routing for each individual subnet on each of the routers they connected to. In order to ensure it was working properly I pinged between the different subnet's to ensure connectivity this can be seen below:

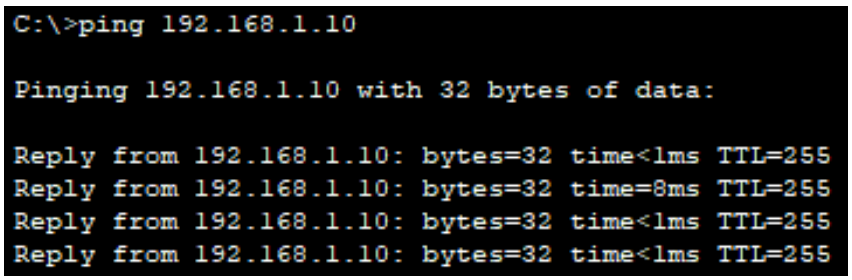

Figure 47: Pinging 192.168.1.10/30 subnet from 10.100.152.0/24 subnet

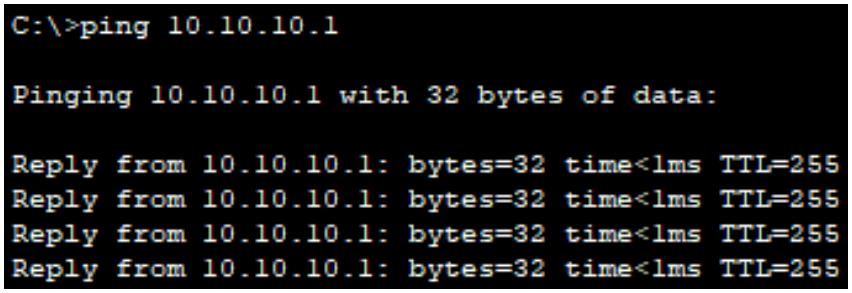

Figure 48: Pinging 10.10.10.1/27 subnet from 192.168.88.0/24 subnet

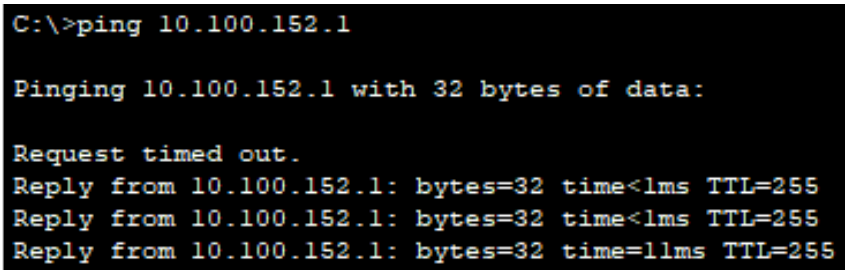

Figure 49: Pinging 10.100.152.0/24 subnet from 192.168.88.1/24 subnet

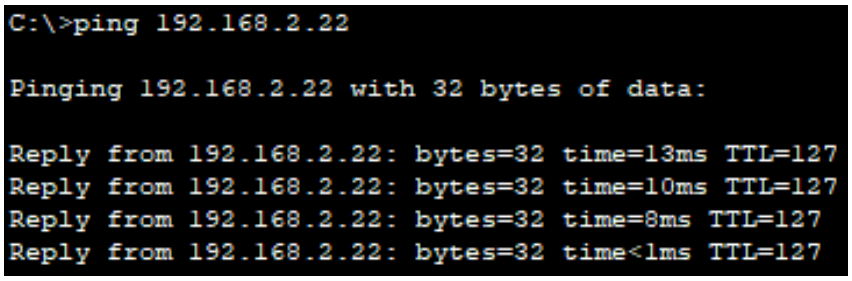

Figure 50: Pinging 192.168.2.0/24 subnet from 192.168.88.1/24 subnet

3 Zones of Trust

3.1 Security Measures

After discovering the topology and creating the network I then moved onto improving the security of the network this involved first giving all routers unique hostnames and enabling passwords in order to prevent unauthorized access. This can be seen below:

```
Router(config)#hostname 10.100.152.1
10.100.152.1(config)#enable secret Pa$$w0rd123
10.100.152.1(config)#line console 0
10.100.152.1(config-line)#password Pa$$w0rd123
10.100.152.1(config-line)#login
10.100.152.1(config-line)#exit
```

Figure 51: Enabling passwords on Routers

After this I moved onto enabling SSH (Secure Shell protocol) on each of the routers to allow for easier changes to router configuration remotely this can be seen below:

```
10.100.152.1(config)#ip domain-name network.com
10.100.152.1(config)#crypto key generate rsa
The name for the keys will be: 10.100.152.1.network.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...[OK]

10.100.152.1(config)#username admin password P4$$word
*Mar 1 2:59:23.282: %SSH-5-ENABLED: SSH 2 has been enabled
10.100.152.1(config)#ip ssh version 2
10.100.152.1(config)#ip ssh authentication-retries 3
10.100.152.1(config)#line vty 0 4
10.100.152.1(config-line)#login local
10.100.152.1(config-line)#transport input ssh
10.100.152.1(config-line)#end
```

Figure 52: Enabling SSH on Routers

In order to test whether this was working as intended I connected to the 192.168.88.1 router as seen below:

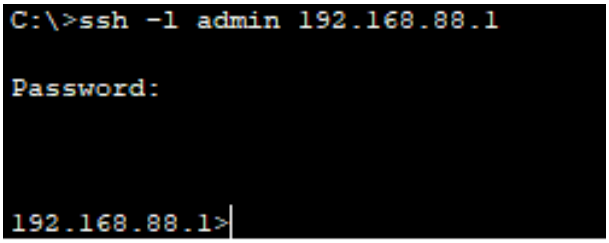

Figure 53: Testing SSH on Routers

3.2 Zone policy firewall

When creating the Zone policy firewall I decided to create three different zones one for the LAN, one for the DMZ and one for the WAN. This system of zoning will help protect the internal network from external threats by blocking incoming unauthorized traffic. Additionally the 192.168.2.0/24 will act as a buffer between the WAN and the LAN and flag any suspicious traffic.

3.3 Zone-Policy Firewall Configuration

In order to begin creating the ZPF configuration I created three zones DMZ, WAN and LAN as seen below:

```
192.168.88.1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
192.168.88.1(config)#zone security LAN
192.168.88.1(config-sec-zone)#zone security WAN
192.168.88.1(config-sec-zone)#zone security DMZ
192.168.88.1(config-sec-zone)#exit
```

Figure 54: Defining ZPF zones

After defining the zones I then moved onto assigning the correct interface to the correct zone this will ensure that traffic cant flow between unauthorized zone. This can be seen below:

```
192.168.88.1(config)#interface GigabitEthernet0/0
192.168.88.1(config-if)#zone-member security LAN
192.168.88.1(config-if)#exit
192.168.88.1(config)#interface GigabitEthernet0/1
192.168.88.1(config-if)#zone-member security WAN
192.168.88.1(config-if)#exit
192.168.88.1(config)#interface GigabitEthernet0/2
192.168.88.1(config-if)#zone-member security DMZ
192.168.88.1(config-if)#exit
192.168.88.1(config)#interface Serial0/0/0
192.168.88.1(config-if)#zone-member security LAN
192.168.88.1(config-if)#exit
192.168.88.1(config)#interface Serial0/0/1
192.168.88.1(config-if)#zone-member security LAN
192.168.88.1(config-if)#exit
```

Figure 55: Assigning interfaces to specific zones

Once the correct interfaces had been assigned I then moved onto defining the zone pairs which dictate where traffic flows too and from. This can be seen below:

```
192.168.88.1(config)#zone-pair security DMZ_to_LAN source DMZ destination LAN
192.168.88.1(config-sec-zone-pair)#service-policy type inspect DMZ_to_LAN
192.168.88.1(config-sec-zone-pair)#
192.168.88.1(config-sec-zone-pair)#zone-pair security LAN_to_DMZ source LAN destination
DMZ
192.168.88.1(config-sec-zone-pair)#service-policy type inspect LAN_to_DMZ
192.168.88.1(config-sec-zone-pair)#
192.168.88.1(config-sec-zone-pair)#zone-pair security WAN_to_DMZ source WAN destination
DMZ
192.168.88.1(config-sec-zone-pair)#service-policy type inspect WAN_to_DMZ
192.168.88.1(config-sec-zone-pair)#
192.168.88.1(config-sec-zone-pair)#zone-pair security DMZ_to_WAN source DMZ destination
WAN
192.168.88.1(config-sec-zone-pair)#service-policy type inspect DMZ_to_WAN
192.168.88.1(config-sec-zone-pair)#
192.168.88.1(config-sec-zone-pair)#zone-pair security WAN_to_LAN source WAN destination
LAN
192.168.88.1(config-sec-zone-pair)#service-policy type inspect WAN_to_LAN
192.168.88.1(config-sec-zone-pair)#
192.168.88.1(config-sec-zone-pair)#zone-pair security LAN_to_WAN source LAN destination
WAN
192.168.88.1(config-sec-zone-pair)#service-policy type inspect LAN_to_WAN
```

Figure 56: Defining zone pairs

With this step completed I was then able to create class maps which allow certain types of traffic between the devices within the networks. I restricted it just to UDP, TCP and ICMP however, this can easily be expanded to accommodate more protocols. This can be seen below:

```
192.168.88.1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
192.168.88.1(config)#class-map type inspect match-any ALL_TRAFFIC
192.168.88.1(config-cmap)# match protocol tcp
192.168.88.1(config-cmap)# match protocol udp
192.168.88.1(config-cmap)# match protocol icmp
192.168.88.1(config-cmap)#exit
```

Figure 57: Filtering traffic to only allow certain protocols

Finally, in order to complete the zone policy firewall I applied the policy maps to the zone pairs this will ensure that traffic is properly filtered. This can be seen below:

```
192.168.88.1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
192.168.88.1(config)#class-map type inspect match-any ALL_TRAFFIC
192.168.88.1(config-cmap)#match protocol ip
192.168.88.1(config-cmap)#exit
192.168.88.1(config)#policy-map type inspect DMZ_to_LAN
192.168.88.1(config-pmap)#class type inspect ALL_TRAFFIC
192.168.88.1(config-pmap-c)#pass
192.168.88.1(config-pmap-c)#exit
192.168.88.1(config-pmap)#
192.168.88.1(config-pmap)#policy-map type inspect LAN_to_DMZ
192.168.88.1(config-pmap)#class type inspect ALL_TRAFFIC
192.168.88.1(config-pmap-c)#pass
192.168.88.1(config-pmap-c)#exit
192.168.88.1(config-pmap)#
192.168.88.1(config-pmap)#policy-map type inspect WAN_to_DMZ
192.168.88.1(config-pmap)#class type inspect ALL_TRAFFIC
192.168.88.1(config-pmap-c)#pass
192.168.88.1(config-pmap-c)#exit
192.168.88.1(config-pmap)#
192.168.88.1(config-pmap)#policy-map type inspect DMZ_to_WAN
192.168.88.1(config-pmap)#class type inspect ALL_TRAFFIC
192.168.88.1(config-pmap-c)#pass
192.168.88.1(config-pmap-c)#exit
192.168.88.1(config-pmap)#
192.168.88.1(config-pmap)#policy-map type inspect WAN_to_LAN
192.168.88.1(config-pmap)#class class-default
192.168.88.1(config-pmap-c)#drop
192.168.88.1(config-pmap-c)#exit
192.168.88.1(config-pmap)#
192.168.88.1(config-pmap)#policy-map type inspect LAN_to_WAN
192.168.88.1(config-pmap)#class type inspect ALL_TRAFFIC
192.168.88.1(config-pmap-c)#pass
192.168.88.1(config-pmap-c)#exit
```

Figure 58: Applying policy maps to individual zone pairs

3.4 Connectivity Verification

In order to ensure it was working as intended I tested zone to zone transmissions by pinging between devices in each different zone. The results can be seen below:

```
C:\>ping 192.168.88.30

Pinging 192.168.88.30 with 32 bytes of data:

Reply from 192.168.88.30: bytes=32 time=1ms TTL=127
Reply from 192.168.88.30: bytes=32 time=8ms TTL=127
Reply from 192.168.88.30: bytes=32 time<1ms TTL=127
Reply from 192.168.88.30: bytes=32 time=2ms TTL=127
```

Figure 59: Successful ping from DMZ to LAN

```
C:\>ping 192.168.2.22

Pinging 192.168.2.22 with 32 bytes of data:

Reply from 192.168.2.22: bytes=32 time=9ms TTL=127
Reply from 192.168.2.22: bytes=32 time<1ms TTL=127
Reply from 192.168.2.22: bytes=32 time<1ms TTL=127
Reply from 192.168.2.22: bytes=32 time<1ms TTL=127
```

Figure 60: Successful ping from external server to DMZ

```
C:\>ping 192.168.88.30

Pinging 192.168.88.30 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

Figure 61: Unsuccessful ping from external server to LAN

These tests show that the ZPF is working as intended as devices in the LAN cant communicate with devices in the WAN however, devices in the DMZ can communicate with both.

4 Conclusion

In conclusion, through the analysis of packets with tools such as Wireshark and NetworkMiner I was able to enhance my understanding of the network infrastructure. This enabled me to recreate the Network in Cisco packet tracer complete with all its services and the correct IPV4 addresses. This first stage also aided me in proposing a more secure network design which utilized features such as Zone Policy Firewalls and SSH. The successful implementation of these features is further highlighted by the blocked connections between zones that aren't meant to communicate. This report provides an in depth analysis of the network while also providing a number of recommendations to increase its security.

References

[1] wiki.wireshark.org. (n.d.). FrontPage - The Wireshark Wiki. [online] Available at: https://wiki.wireshark.org.

[2] wiki.wireshark.org. (n.d.). S7comm - The Wireshark Wiki. [online] Available at: https://wiki.wireshark.org/S7comm.

[3] GeeksforGeeks. (2018). Simple Network Management Protocol (SNMP) - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/simple-network-management-protocol-snmp/.

[4] Prashant Lakhera (2017). HTTP/HTTPS Analysis Using Wireshark. [online] Medium. Available at: https://medium.com/devops-world/http-https-analysis-using-wireshark-cbe07c23520.

[5] learningnetwork.cisco.com. (n.d.). Cisco Learning Network. [online] Available at: https://learningnetwork.cisco.com/s/question/0D53i00000KsusdCAB/zonebased-policy-firewalls-5-step-process.