

Portfolio Project

Packet Analysis and Simple DoS using Wireshark And Hping

**Ranilo John M. Delos Angeles
University of the East - Caloocan**

I. Introduction

From learning the topology creations in Cisco Packet Tracer, I wanted to use a more complicated network simulator and so in this scenario, I simulated topology where I can access the computers through Virtual Machines and this was made possible by GNS3 and Virtual Box.

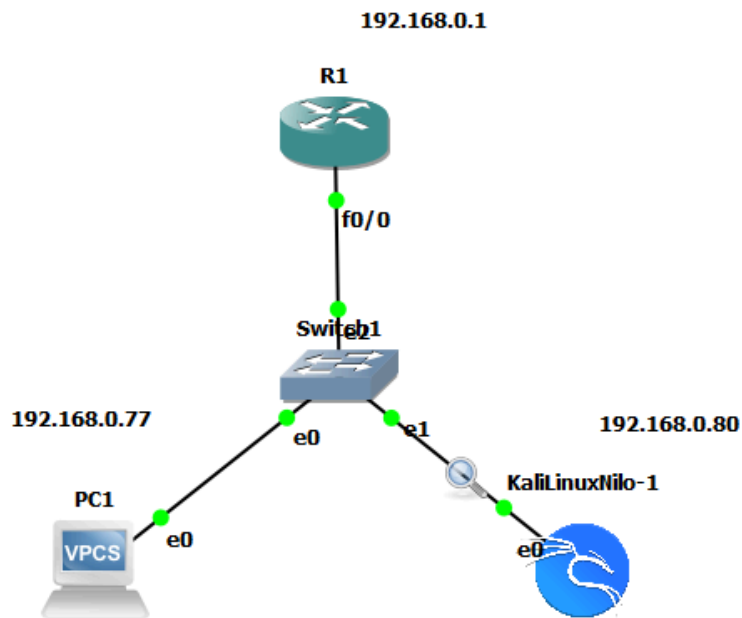
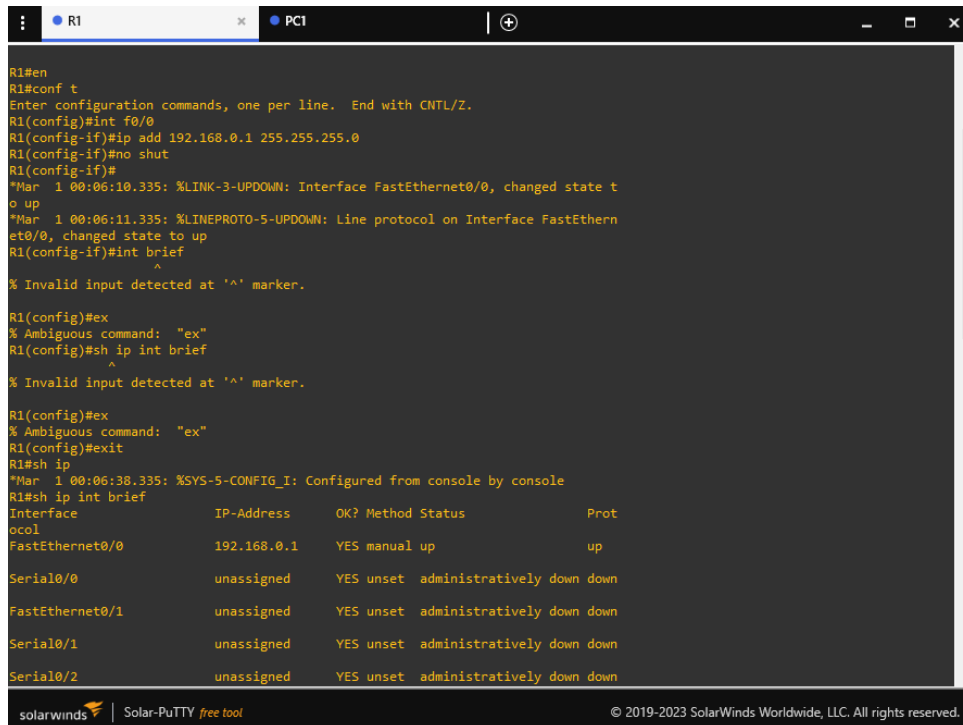


Figure 1.1 GNS3 Simple Topology

The devices I used was a Cisco Router C3745, a Simple Virtual PC, Ethernet Switch, and a Kali Linux simulated using VirtualBox.

II. Configuration

I first started configuring the router, giving the switch its IP Address of 192.168.0.1 255.255.255.0. It is an IP Address because I wanted to check first how I can properly configure and simulate the network. I also went to configure the PC1 with the IP Address of 192.168.0.77 and Kali Linux VM IP Address of 192.168.0.77



```
R1#en
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#int f0/0
R1(config-if)#ip add 192.168.0.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#
*Mar 1 00:06:10.335: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state t
o up
*Mar 1 00:06:11.335: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthern
et0/0, changed state to up
R1(config-if)#int brief
^
% Invalid input detected at '^' marker.

R1(config)#ex
% Ambiguous command: "ex"
R1(config)#sh ip int brief
^
% Invalid input detected at '^' marker.

R1(config)#ex
% Ambiguous command: "ex"
R1(config)#exit
R1#sh ip
*Mar 1 00:06:38.335: %SYS-5-CONFIG_I: Configured from console by console
R1#sh ip int brief
Interface IP-Address OK? Method Status Prot
ocol
FastEthernet0/0 192.168.0.1 YES manual up up
Serial0/0 unassigned YES unset administratively down down
FastEthernet0/1 unassigned YES unset administratively down down
Serial0/1 unassigned YES unset administratively down down
Serial0/2 unassigned YES unset administratively down down
```

Figure 2.1 Router IP Address

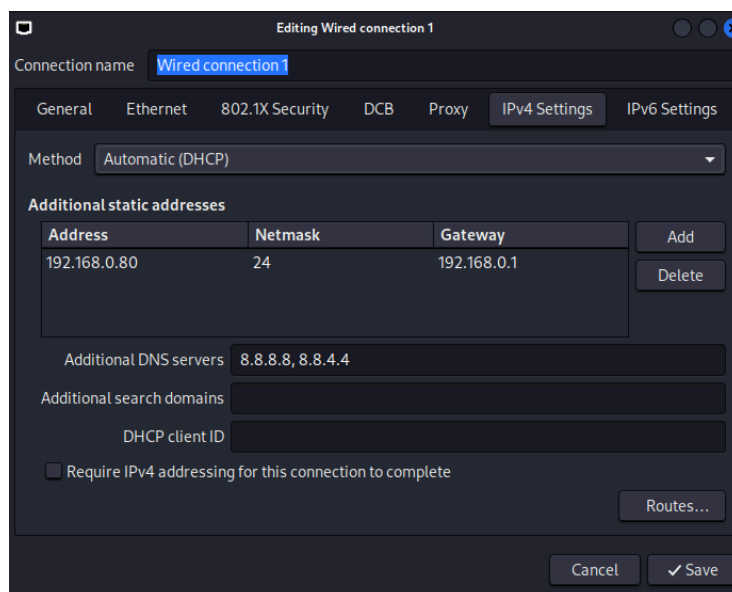
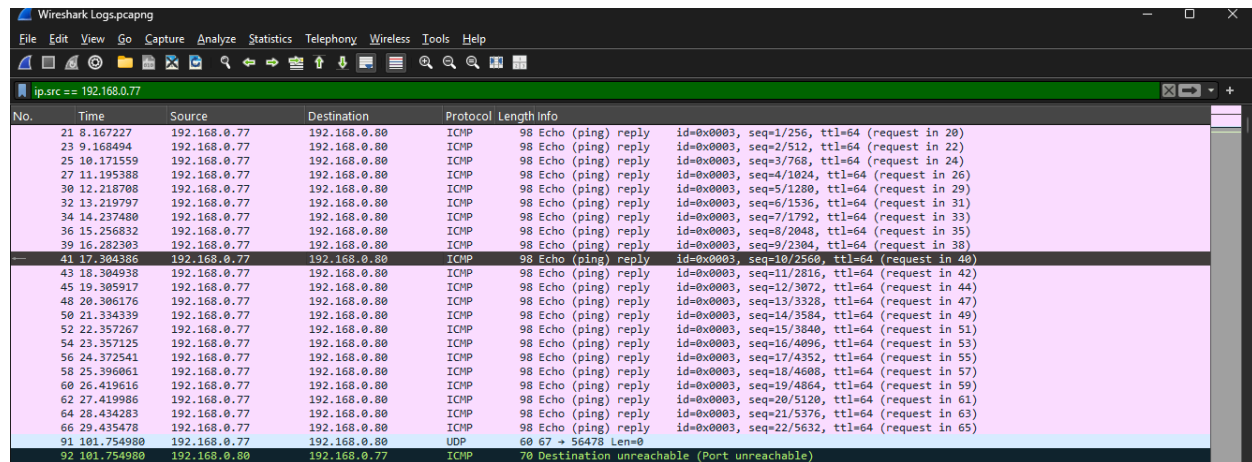


Figure 2.2 Kali Linux IP Address

III. Simulation

So first from the VPCS, I went to ICMP ping the destination IP of 192.168.0.80 (Kali Linux PC) for error checking (to see if it works or not)

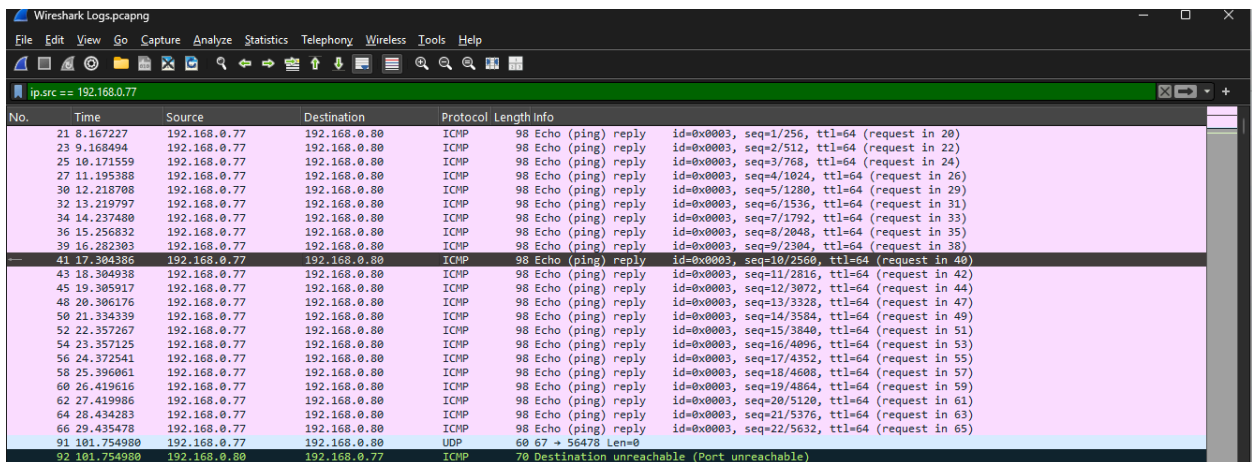


The image shows a Wireshark capture of ICMP Echo (ping) requests and replies from source IP 192.168.0.77 to destination IP 192.168.0.80. The capture shows a series of successful pings from packet 21 to 40, followed by a 'Destination unreachable (Port unreachable)' message at packet 70. The filter is set to 'ip.src == 192.168.0.77'.

No.	Time	Source	Destination	Protocol	Length	Info
21	8.167227	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=1/256, ttl=64 (request in 20)
23	9.168494	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=2/512, ttl=64 (request in 22)
25	10.171559	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=3/768, ttl=64 (request in 24)
27	11.195388	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=4/1024, ttl=64 (request in 26)
30	12.218708	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=5/1280, ttl=64 (request in 29)
32	13.219797	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=6/1536, ttl=64 (request in 31)
34	14.237480	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=7/1792, ttl=64 (request in 33)
36	15.256832	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=8/2048, ttl=64 (request in 35)
39	16.282303	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=9/2304, ttl=64 (request in 38)
41	17.304386	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=10/2560, ttl=64 (request in 40)
43	18.304938	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=11/2816, ttl=64 (request in 42)
45	19.305917	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=12/3072, ttl=64 (request in 44)
48	20.306176	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=13/3328, ttl=64 (request in 47)
50	21.334339	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=14/3584, ttl=64 (request in 49)
52	22.357267	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=15/3840, ttl=64 (request in 51)
54	23.357125	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=16/4096, ttl=64 (request in 53)
56	24.372541	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=17/4352, ttl=64 (request in 55)
58	25.396061	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=18/4608, ttl=64 (request in 57)
60	26.419616	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=19/4864, ttl=64 (request in 59)
62	27.419986	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=20/5120, ttl=64 (request in 61)
64	28.434283	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=21/5376, ttl=64 (request in 63)
66	29.435478	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=22/5632, ttl=64 (request in 65)
91	101.754980	192.168.0.77	192.168.0.80	UDP	60	67 → 56478 Len=0
92	101.754980	192.168.0.80	192.168.0.77	ICMP	70	Destination unreachable (Port unreachable)

Figure 3.1 ICMPing VPCS to Kali Linux

I also did the same for Kali, I ICMP ping from Kali with IP Address of 192.168.0.80 to VPCS 192.168.0.77.



The image shows a Wireshark capture of ICMP Echo (ping) requests and replies from source IP 192.168.0.80 to destination IP 192.168.0.77. The capture shows a series of successful pings from packet 21 to 40, followed by a 'Destination unreachable (Port unreachable)' message at packet 70. The filter is set to 'ip.src == 192.168.0.77'.

No.	Time	Source	Destination	Protocol	Length	Info
21	8.167227	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=1/256, ttl=64 (request in 20)
23	9.168494	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=2/512, ttl=64 (request in 22)
25	10.171559	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=3/768, ttl=64 (request in 24)
27	11.195388	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=4/1024, ttl=64 (request in 26)
30	12.218708	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=5/1280, ttl=64 (request in 29)
32	13.219797	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=6/1536, ttl=64 (request in 31)
34	14.237480	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=7/1792, ttl=64 (request in 33)
36	15.256832	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=8/2048, ttl=64 (request in 35)
39	16.282303	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=9/2304, ttl=64 (request in 38)
41	17.304386	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=10/2560, ttl=64 (request in 40)
43	18.304938	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=11/2816, ttl=64 (request in 42)
45	19.305917	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=12/3072, ttl=64 (request in 44)
48	20.306176	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=13/3328, ttl=64 (request in 47)
50	21.334339	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=14/3584, ttl=64 (request in 49)
52	22.357267	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=15/3840, ttl=64 (request in 51)
54	23.357125	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=16/4096, ttl=64 (request in 53)
56	24.372541	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=17/4352, ttl=64 (request in 55)
58	25.396061	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=18/4608, ttl=64 (request in 57)
60	26.419616	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=19/4864, ttl=64 (request in 59)
62	27.419986	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=20/5120, ttl=64 (request in 61)
64	28.434283	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=21/5376, ttl=64 (request in 63)
66	29.435478	192.168.0.77	192.168.0.80	ICMP	98	Echo (ping) reply id=0x0003, seq=22/5632, ttl=64 (request in 65)
91	101.754980	192.168.0.77	192.168.0.80	UDP	60	67 → 56478 Len=0
92	101.754980	192.168.0.80	192.168.0.77	ICMP	70	Destination unreachable (Port unreachable)

Figure 3.2 ICMPing Kali to VPCS

After that, I started doing nbtscan (netbios scan) on the network to check the ip addresses.

```
(root@kali)-[/home/kali]
# nbtscan -r 192.168.0.1/24
Doing NBT name scan for addresses from 192.168.0.1/24
```

IP address	NetBIOS Name	Server	User	MAC address
192.168.0.80	<unknown>		<unknown>	
192.168.0.77	<unknown>		<unknown>	

```
^C
```

Figure 3.3 NBT scan in the network.

Finishing the nbtscan, I went ahead and used the NMAP command for the scanning of the VPCS, the report below shows the open ports (All ports are open because the PC is unsecured) the -sS does this by getting a SYN/ACK reply and then gets to RST to not complete the three way handshake.

```
root@kali: /home/kali
Session Actions Edit View Help
(kali@kali)-[~]
$ nmap -sS 192.168.0.77
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-02 08:25 EST
Nmap scan report for 192.168.0.77
Host is up (0.0016s latency).
```

PORT	STATE	SERVICE
1/tcp	open	tcpmux
3/tcp	open	compressnet
4/tcp	open	unknown
6/tcp	open	unknown
7/tcp	open	echo
9/tcp	open	discard
13/tcp	open	daytime
17/tcp	open	qotd
19/tcp	open	chargen
20/tcp	open	ftp-data
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
24/tcp	open	priv-mail
25/tcp	open	smtp
26/tcp	open	rsftp
30/tcp	open	unknown
32/tcp	open	unknown
33/tcp	open	dsp
37/tcp	open	time

Figure 3.4 NMAP the ports to check which is open.

Here below, shows all the captured packet from the attacker's machine (Kali) the displayed packet are 2018. It shows all the ports from 1 to 65389. It also shows the MAC Address of the VPCS, 00:50:79:66:68:00.



Figure 3.5 NMAP scan finished.

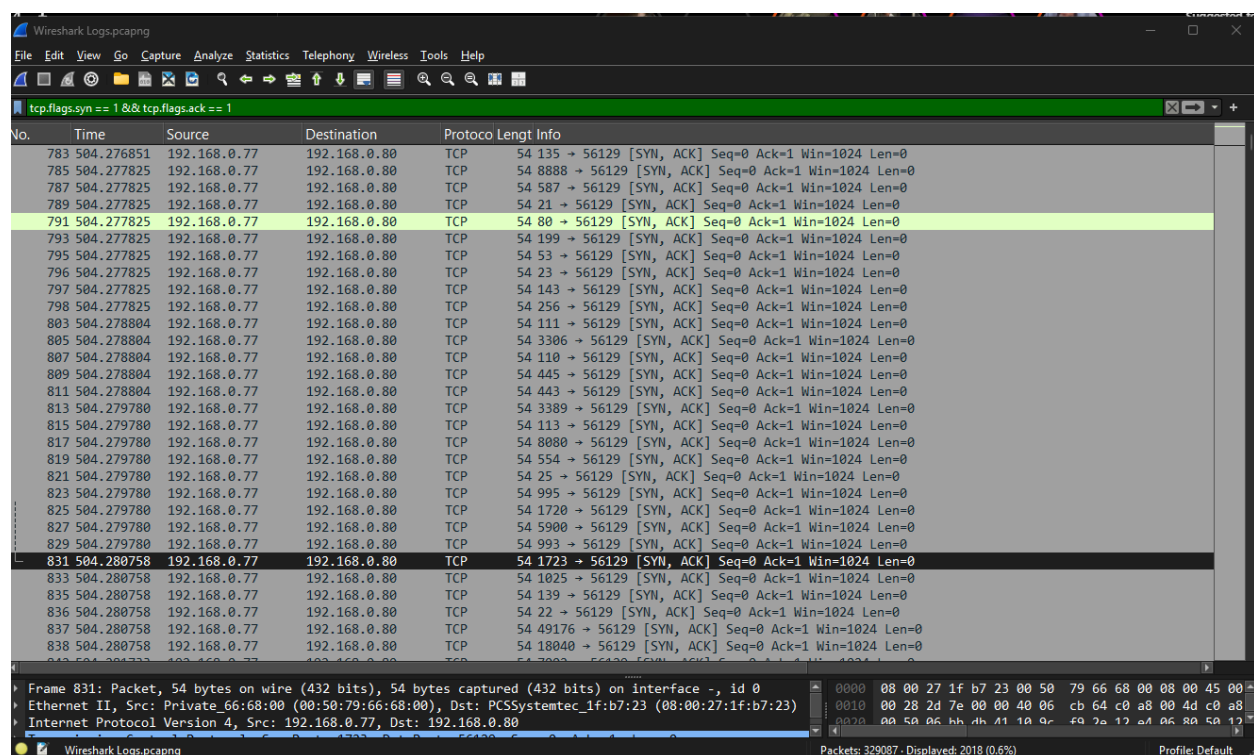
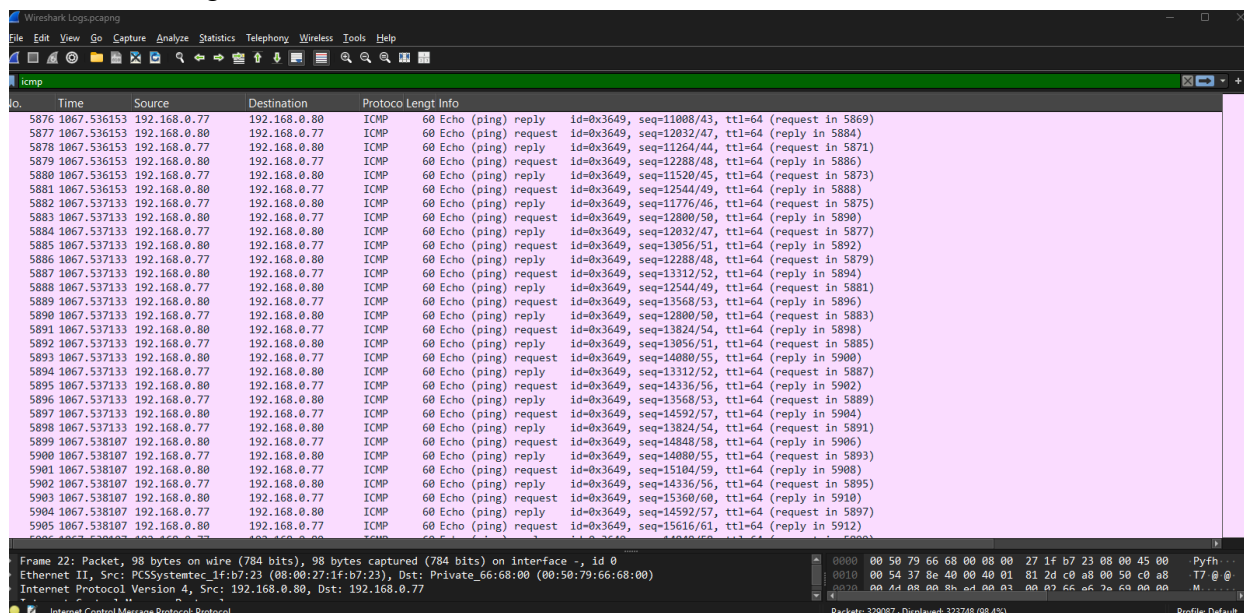


Figure 3.6 Packets captured from NMAP scanning from Port 1 to 65389.

Now since I'm curious about Denial of Service, I tried flooding the VPCS with command `hping3 -icmp -flood 192.168.0.77`, As shown below, there's about 245665 packets transmitted to the other end device it also did put the PC into a session timeout due to overwhelming amounts of ICMP.



No.	Time	Source	Destination	Protocol	Length	Info
5876	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=11080/43, ttl=64 (request in 5869)
5877	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12032/47, ttl=64 (request in 5884)
5878	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=11264/44, ttl=64 (request in 5871)
5879	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12288/48, ttl=64 (request in 5886)
5880	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=11520/45, ttl=64 (request in 5873)
5881	1067.536153	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12544/49, ttl=64 (request in 5888)
5882	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=11776/46, ttl=64 (request in 5875)
5883	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12800/50, ttl=64 (request in 5890)
5884	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12032/47, ttl=64 (request in 5877)
5885	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13056/51, ttl=64 (request in 5892)
5886	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12288/48, ttl=64 (request in 5879)
5887	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13312/52, ttl=64 (request in 5894)
5888	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12544/49, ttl=64 (request in 5881)
5889	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13568/53, ttl=64 (request in 5896)
5890	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=12800/50, ttl=64 (request in 5883)
5891	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13824/54, ttl=64 (request in 5898)
5892	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13056/51, ttl=64 (request in 5885)
5893	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14080/55, ttl=64 (request in 5900)
5894	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13312/52, ttl=64 (request in 5887)
5895	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14336/56, ttl=64 (request in 5902)
5896	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13568/53, ttl=64 (request in 5889)
5897	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14592/57, ttl=64 (request in 5904)
5898	1067.537133	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=13824/54, ttl=64 (request in 5891)
5899	1067.538107	192.168.0.80	192.168.0.77	ICMP	60	Echo (ping) request id=0x3649, seq=14848/58, ttl=64 (request in 5906)
5900	1067.538107	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14080/55, ttl=64 (request in 5893)
5901	1067.538107	192.168.0.80	192.168.0.77	ICMP	60	Echo (ping) request id=0x3649, seq=15104/59, ttl=64 (request in 5908)
5902	1067.538107	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14336/56, ttl=64 (request in 5895)
5903	1067.538107	192.168.0.80	192.168.0.77	ICMP	60	Echo (ping) request id=0x3649, seq=15360/60, ttl=64 (request in 5910)
5904	1067.538107	192.168.0.77	192.168.0.80	ICMP	60	Echo (ping) request id=0x3649, seq=14592/57, ttl=64 (request in 5897)
5905	1067.538107	192.168.0.80	192.168.0.77	ICMP	60	Echo (ping) request id=0x3649, seq=15616/61, ttl=64 (request in 5912)

Figure 3.7 Packet Capture of DoS ICMP Flooding

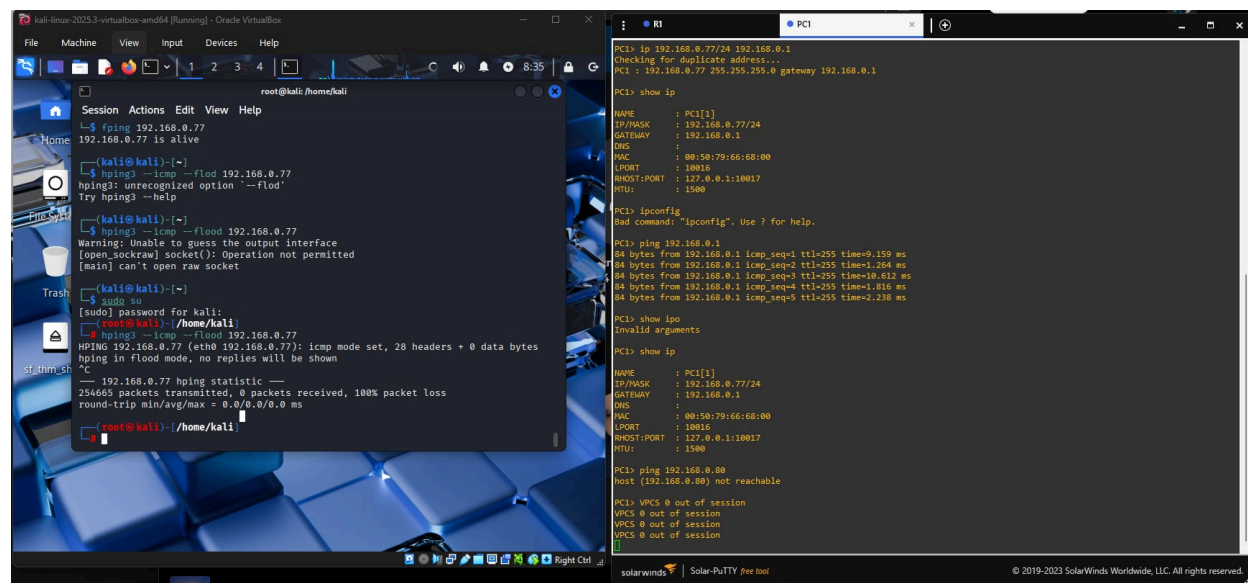


Figure 3.8 Kali command and transmitted packets and putting VPCS down.

IV. Conclusion

This has been a fun experiment to play with, because now I can finally simulate penetration testing via Kali Linux, but for this kind of simulation I want to explore more on how I will defend organizations via IDS and IPS.