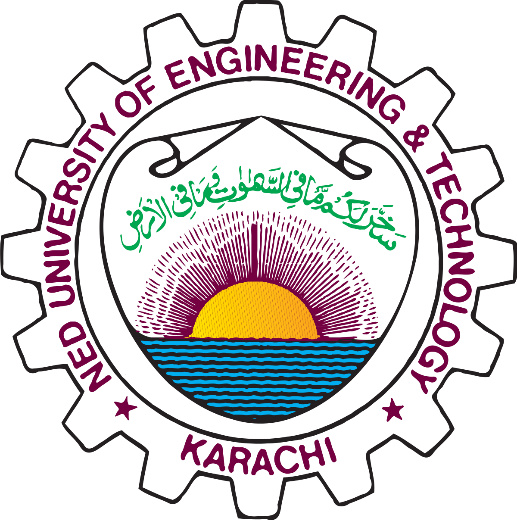
**CT-541 – NETWORK SECURITY**MS-IS 004 2019/20– Evening Fall 2019

**CT-541 NS Assignment-01**

**ARP Spoofing, DoS, Redirection**



**STUDENT NAME: MUHAMMAD UMAR TARIQ**

**ROLL NUMBER : IS 004 2019/20**

**COURSE INSTRUCTOR: Dr. Mubashir M Khan**

**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY**

**NED UNIVERSITY, KARACHI CAMPUS**

**To Demonstrate ARP Cache Poisoning we use netwox 72 command.**

72: Some systems reject the ARP replies if they didn’t ask for it. This is

when this tool comes in handy. It sends an ARP request to the destination

machine using spoofed addresses (Can be used in ARP cache poisoning).

The only issue with this tool is that it does not send continuous packets.

Number of Virtual Machines:

1. Attacker Virtual Machine 1
2. Virtual Machine 2 [Host A]
3. Virtual Machine 2 [Host B]

**IP Configuration of Virtual Machines Before Attack:**

**Attacker , VM1:**

enp0s3 Link encap:Ethernet HWaddr 08:00:27:f8:35:81

inet addr:10.0.2.4 Bcast:10.0.2.255 Mask:255.255.255.0

inet6 addr: fe80::5ada:2e32:bf6a:39a1/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:205 errors:0 dropped:0 overruns:0 frame:0

TX packets:194 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:54623 (54.6 KB) TX bytes:20910 (20.9 KB)

**Host A, VM2**:

[07/09/20]seed@VM:~$ sudo ifconfig

enp0s3 Link encap:Ethernet HWaddr 08:00:27:1b:67:b5

inet addr:10.0.2.5 Bcast:10.0.2.255 Mask:255.255.255.0

inet6 addr: fe80::1b82:22f2:6b00:abcf/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:123 errors:0 dropped:0 overruns:0 frame:0

TX packets:126 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:42813 (42.8 KB) TX bytes:13685 (13.6 KB)

**Host B, VM3**:

enp0s3 Link encap:Ethernet HWaddr 08:00:27:b6:6c:b5

inet addr:10.0.2.6 Bcast:10.0.2.255 Mask:255.255.255.0

inet6 addr: fe80::ef9b:10a3:54da:35d2/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:7 errors:0 dropped:0 overruns:0 frame:0

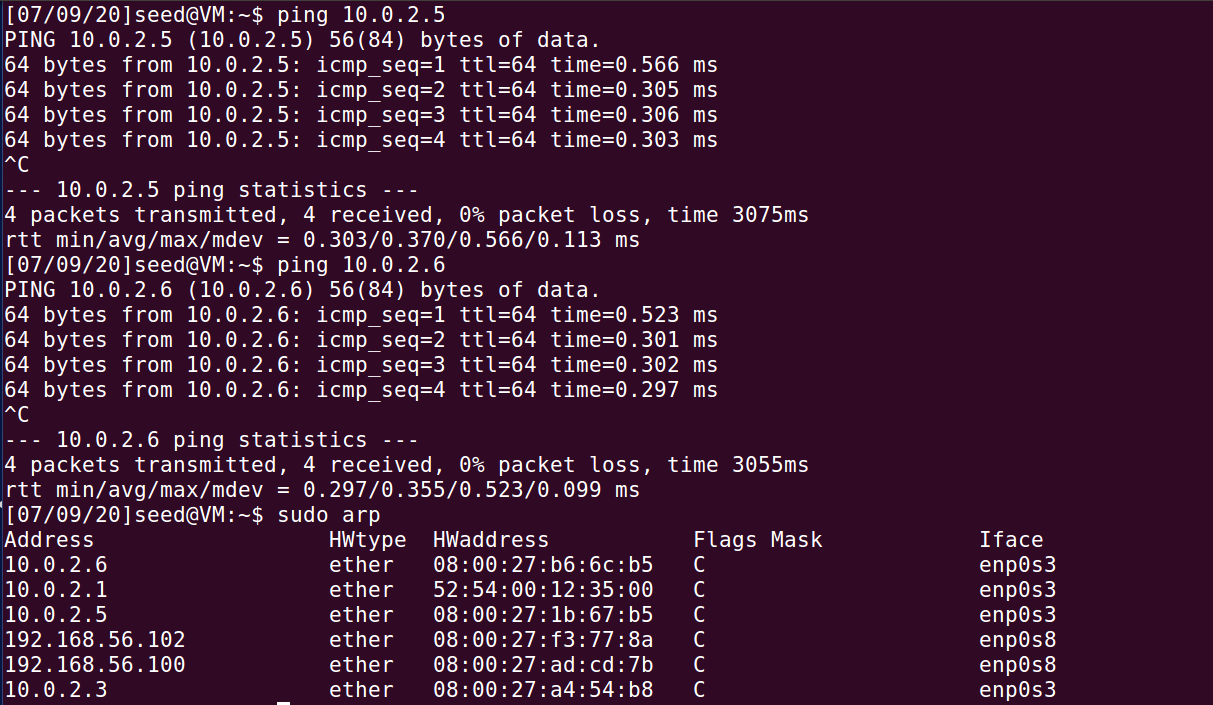
TX packets:67 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

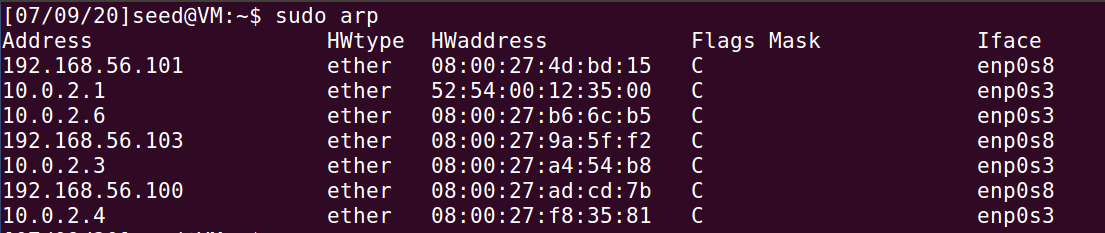
RX bytes:1768 (1.7 KB) TX bytes:7689 (7.6 KB)

**ARP Cache Before Attack:**

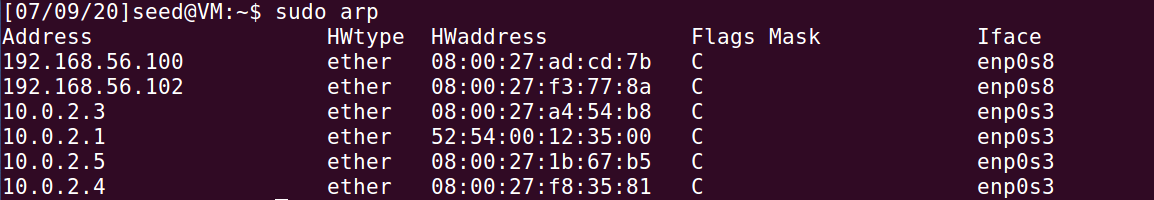
**VM1:**



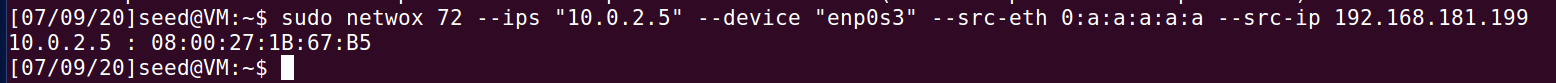
**VM2:**



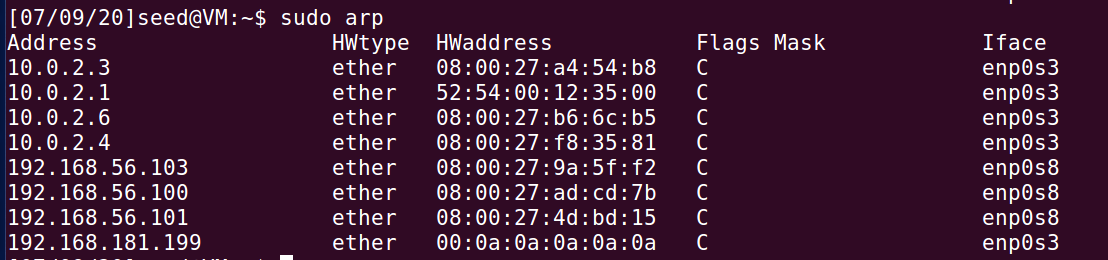
**VM3:**



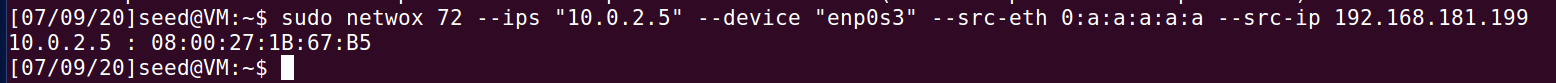
**Scenario 1: Add a Non Existent IP Address and MAC Address in ARP Cache Of Host A i.e VM2**



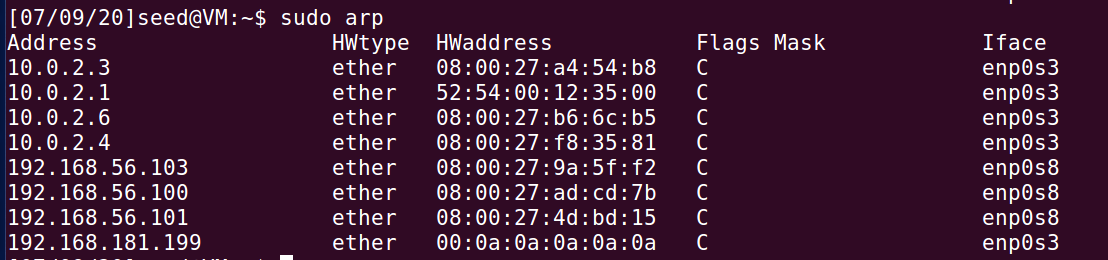
**Result : 192.168.181.199 added with non-existent MAC Address**



**Scenario 2: Add a Non Existent IP Address and MAC Address in ARP Cache Of Host A i.e VM2 For DoS Attack**



**Result : 192.168.181.199 added with non-existent MAC Address**

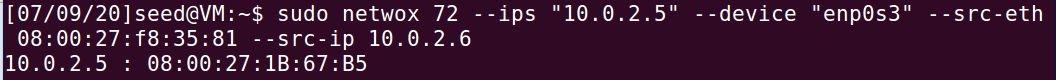


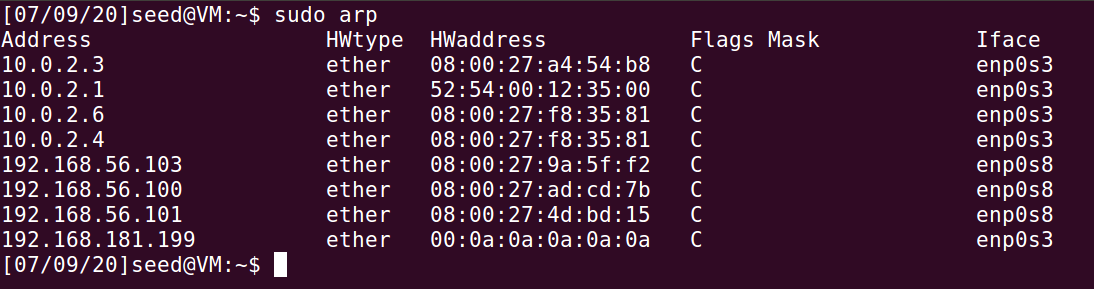
**Scenario 3: Redirect all Traffic between VM2 AND VM3 to VM1:**

* **Associate Attacker(VM1) MAC Address to VM3 IP on VM2**

**FROM VM1 Machine -> On VM2 machine, in ARP cache table, associate VM1 Attackers MAC address to VM3 Ip Address.**

**Result 10.0.2.4 and 10.0.2.6 both have same MAC Address of VM1.**

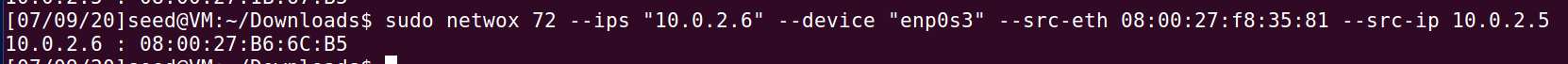


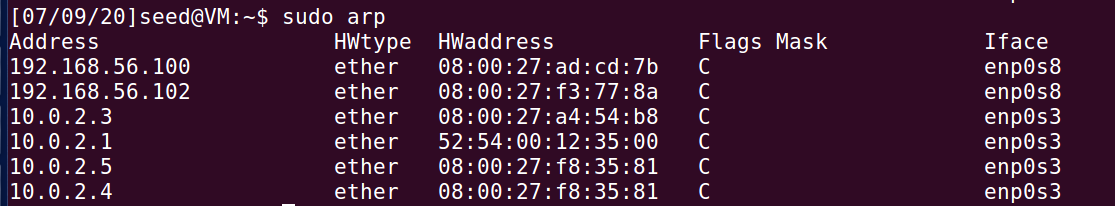


* **Associate Attacker(VM1) MAC Address to VM2 IP on VM3**

**FROM VM1 Machine -> On VM3 machine, in ARP cache table, associate VM1 Attackers MAC address to VM2 IP Address.**

**Result 10.0.2.4 and 10.0.2.5 both have same MAC Address of VM1**





**To Demonstrate Network Traffic Redirection, we use Telnet Client and Server Communication built in to ubuntu.**

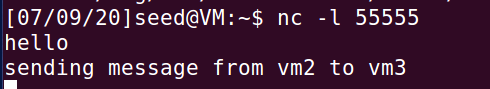
**Machine vm3 is used as telnet server**

**Machine vm2 is used as telnet client**

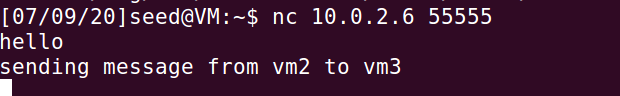
**Keep ip forwarding on when telnet is established on VM1**



**Establish telnet vm3 (ON VM3)**



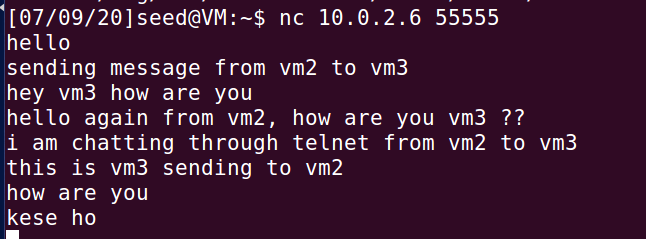
**telnet from vm2 to vm3 (ON VM2)**



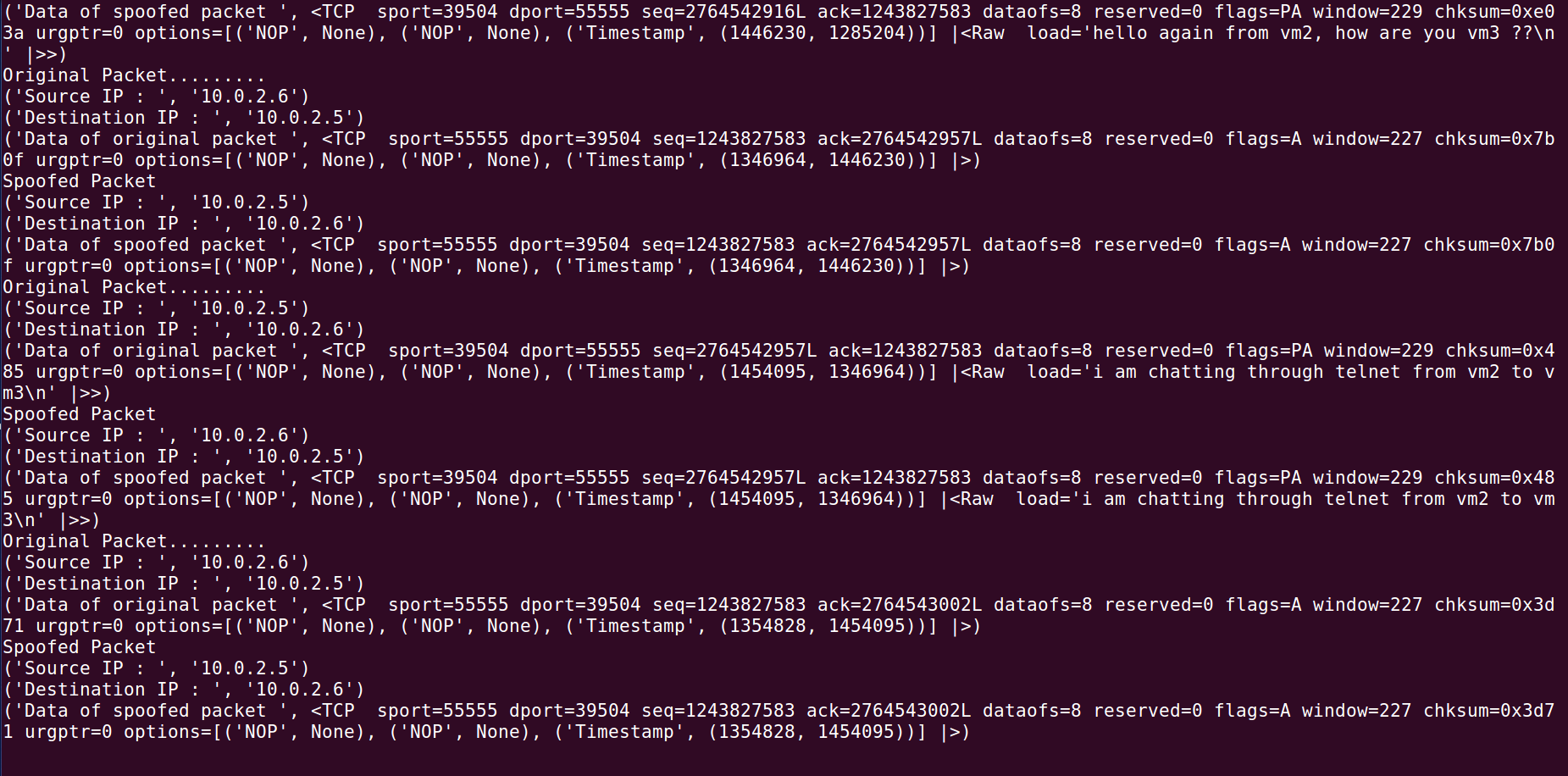
**Turn off ip forwarding (ON VM1)**

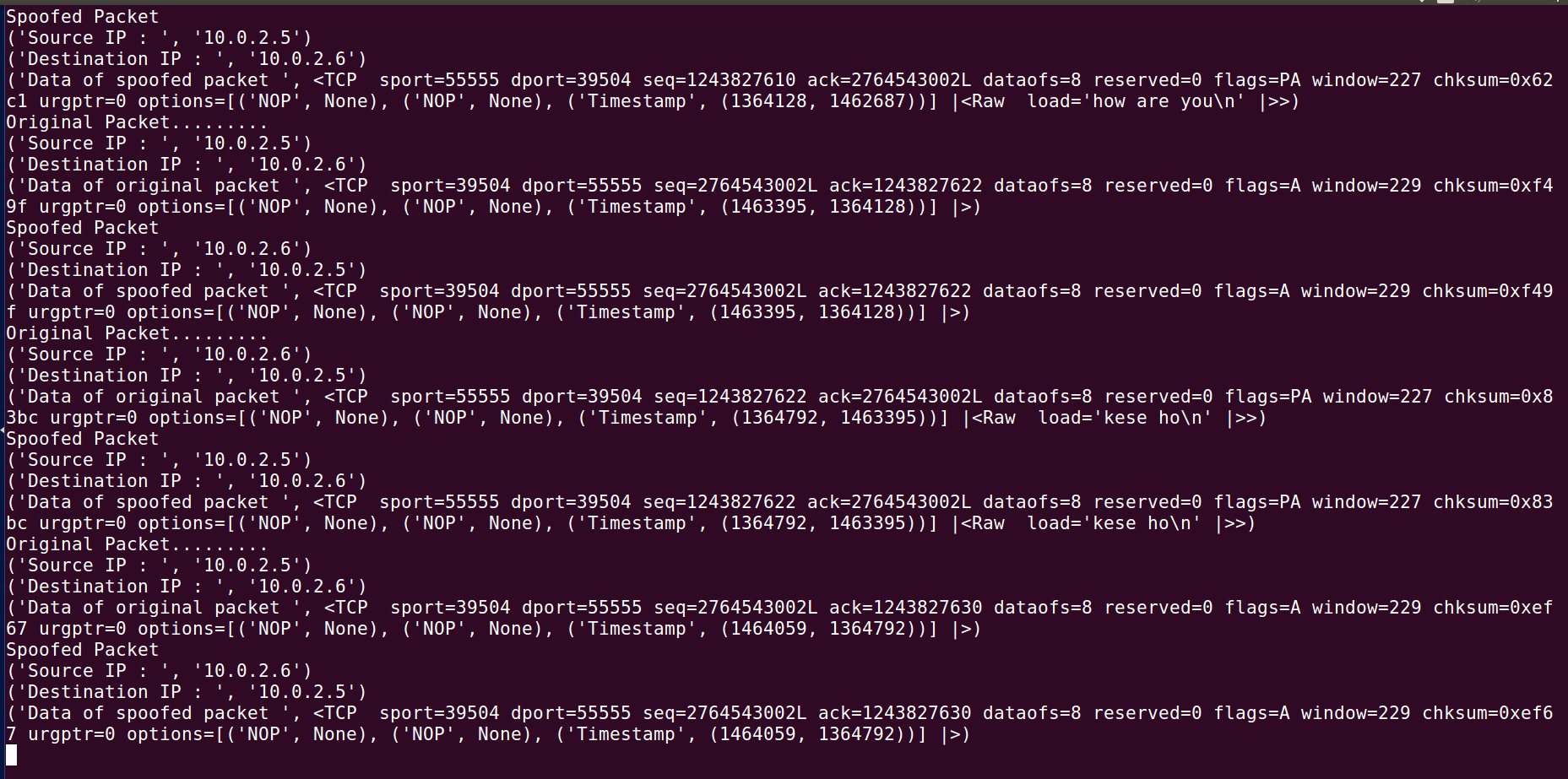


**Telnet Chat from VM2 TO VM3.**



**Output captured on VM1 Terminal. When ip forwarding is off.**





By Running Source Code python script on VM1, we are able to capture packets going between vm2 and vm3:

#!/bin/bin/python

from scapy.all import \*

def spoof\_pkt(pkt):

print("Original Packet.........")

print("Source IP : ",pkt[IP].src)

print("Destination IP : ",pkt[IP].dst)

print("Data of original packet",pkt[IP].payload)

a = IP()

a.src = pkt[IP].dst

a.dst = pkt[IP].src

b = TCP()

data = pkt[TCP].payload

newpkt = a/b/data

print("Spoofed Packet")

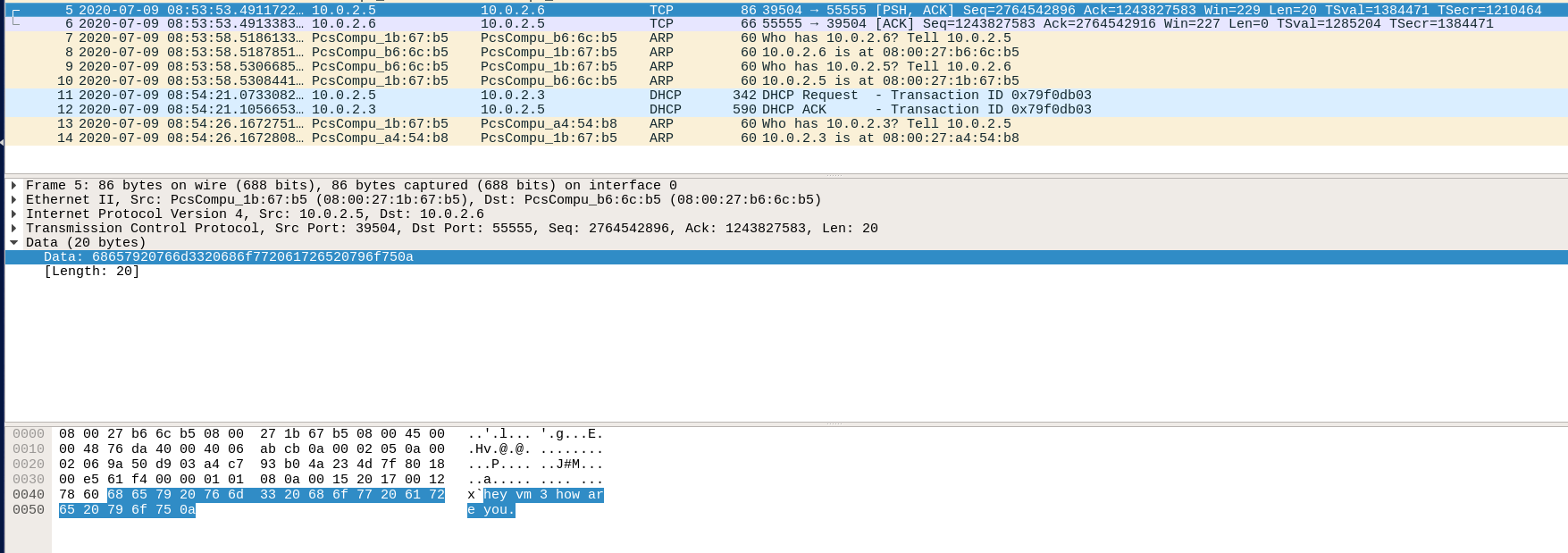
print("Source IP : ",newpkt[IP].src)

print("Destination IP : ",newpkt[IP].dst)

print("Data of spoofed packet",newpkt[IP].payload)

pkt = sniff(filter='tcp', prn=spoof\_pkt)

**WireShark Capture**:

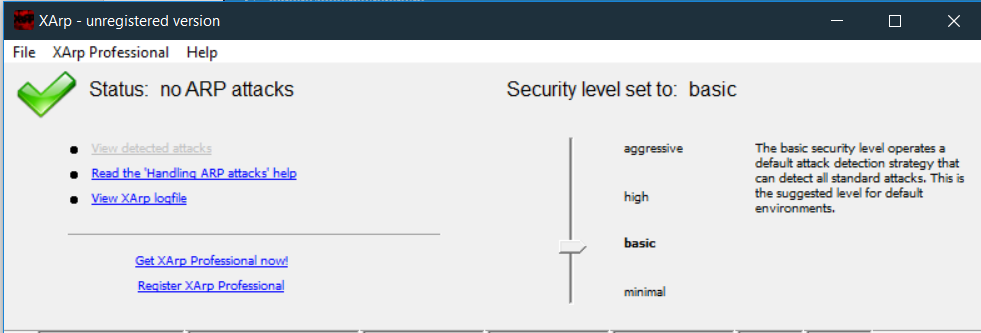


**Observations**: Since the ARP Protocol on Layer 2 has no encryption and authentication we are able to associate ip addresses to MAC address of attacker virtual machine and redirect and deny network traffic to the desktop machines.

**How To Secure Computers Against ARP Attacks:**

**On Windows.**

1. **Software - XArp:**



XArp is a security application that uses advanced techniques to detect ARP based attacks. Using active and passive modules XArp detects hackers inside your network. ARP attacks allow an attacker to silently eavesdrop or manipulate all your data that is sent over the network. This include documents, emails, or VoiceIP conversations. ARP spoofing attacks go undetected by firewalls and operating system security: Firewalls do not protect you against ARP based attack.

1. **On Hardware Switches Through DHCP Snooping** and **Dynamic ARP Inspection**:

DAI has a dependency on DHCP Snooping. In order to run DAI, DHCP Snooping must be enabled.

In computer networking, DHCP snooping is a series of techniques applied to improve the security of a DHCP infrastructure.

When DHCP servers are allocating IP addresses to the clients on the LAN, DHCP snooping can be configured on LAN switches to prevent malicious or malformed DHCP traffic, or rogue DHCP servers. In addition, information on hosts which have successfully completed a DHCP transaction is accrued in a database of "bindings" which may then be used by other security or accounting features.

Other features may use DHCP snooping database information to ensure IP integrity on a Layer 2 switched domain. This information enables a network to:

Track the physical location of IP addresses when combined with AAA accounting or SNMP.

Ensure that hosts only use the IP addresses assigned to them when combined with source-guard a.k.a. source-lockdown

Sanitize ARP requests when combined with arp-inspection a.k.a. arp-protect

Dynamic ARP Inspection (DAI) is a security feature that is available on Cisco Catalyst 6500 Series switches running Cisco IOS Software or Cisco Catalyst OS. Dynamic ARP inspection helps prevent ARP poisoning and other ARP-based attacks by intercepting all ARP (Address Resolution Protocol) requests and responses, and by verifying their authenticity before updating the switch's local ARP cache or forwarding the packets to the intended destinations. Note that on Cisco Catalyst 6500 Series switches, Dynamic ARP requires Supervisor 2, Supervisor 32, or Supervisor 720. As previously stated, a Supervisor 720-3B was used in these tests.

1. On Linux:

Install arpwatch

Use apt-get command under Debian / Ubuntu Linux:

# apt-get install arpwatch

OR

$ sudo apt-get install arpwatch

arpwatch command examples

You can watch particular interface with command:

# arpwatch -i eth0

You will notice syslog entries as follows /var/log/syslog file (or /var/log/message file) when changes are made i.e MAC/IP address pair is changed:

# tail -f /var/log/syslog

References:

1. <http://www.xarp.net/>
2. <https://en.wikipedia.org/wiki/DHCP_snooping>
3. <https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603839.html>
4. <https://www.cyberciti.biz/faq/how-to-detect-arp-spoofing-under-unix-or-linux/>