Unit 4 Assignment-III

Q1. Write a program to prepare TCP and UDP packets using header files and send the packets to destination machine in peer-to-peer mode. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer-to-peer mode. (Write Algorithm, Print Program with comments and Output)

Algorithm / Procedure:

A. UDP Packet Transmission

1. Open Wireshark and start capturing on the *Npcap Loopback Adapter*.
2. Import Scapy modules using `from scapy.all import *`.
3. Construct IP and UDP headers with source and destination IPs and ports.
4. Attach payload using `Raw(load="Hello from Scapy UDP!")`.
5. Combine layers: `packet = ip / udp / data`.
6. Transmit the packet using `send(packet)`.
7. View and analyze in Wireshark using filter `udp.port == 9090`.

B. TCP SYN Packet Transmission

1. Start Wireshark capture on the *Npcap Loopback Adapter*.
2. Import necessary Scapy modules.
3. Create IP and TCP headers with `flags="S"` for SYN.
4. Combine layers: `syn_pkt = ip / tcp`.
5. Send packet using `send(syn_pkt)`.
6. Observe results using filter `tcp.flags.syn == 1 && tcp.flags.ack == 0`.

Program Code:

(a) UDP Packet (Scapy)

```
from scapy.all import *

# Define IP layer with source and destination IP addresses
# Both IPs are the same for local testing (loopback)
ip = IP(src="192.168.204.107", dst="192.168.204.107")

# Define UDP layer
# sport = Source Port, dport = Destination Port
udp = UDP(sport=8080, dport=9090)
```

```
# Define the payload (data section)
data = Raw(load="Hello from Scapy UDP!")

# Combine all layers to form a complete packet
packet = ip / udp / data

# Display the details of the crafted packet
packet.show()

# Send the packet at layer 3 (network layer)
send(packet)
```

(b) TCP SYN Packet (Scapy)

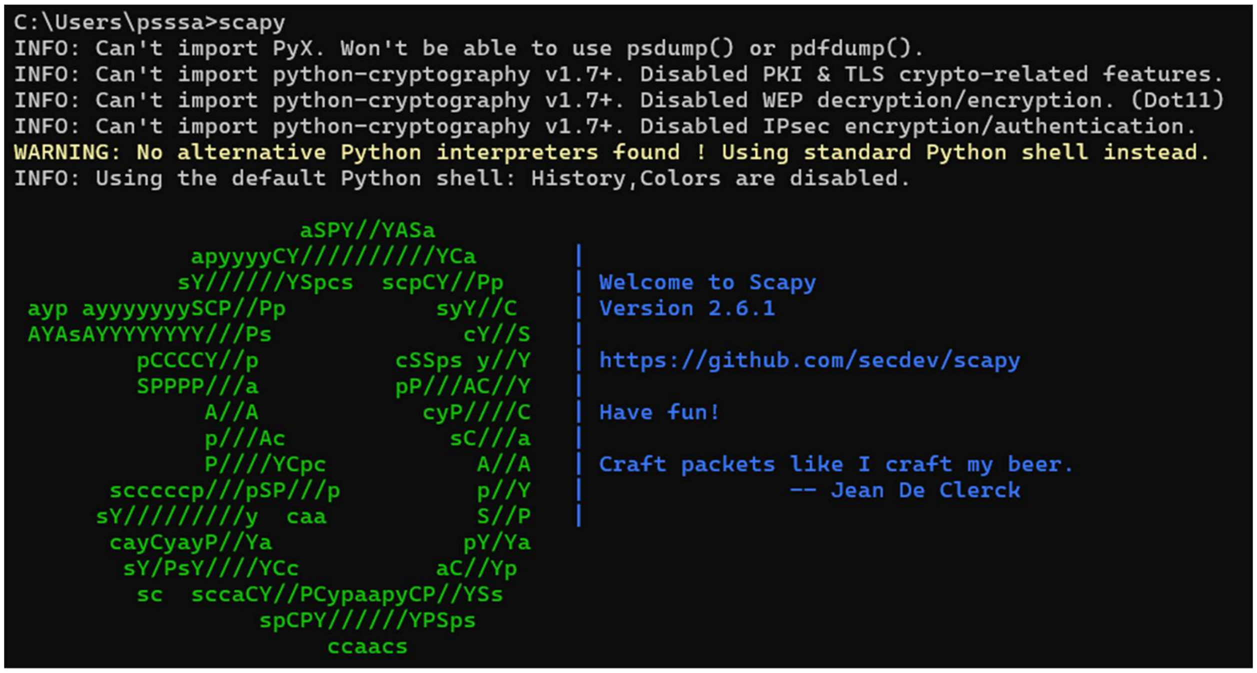
```
from scapy.all import *

# Define IP layer (source and destination are same for local test)
ip = IP(src="192.168.204.107", dst="192.168.204.107")

# Define TCP layer
# sport = Source port
# dport = Destination port (here 8080)
# flags="S" sets the SYN flag (for handshake initiation)
# seq=1000 defines sequence number
tcp = TCP(sport=50000, dport=8080, flags="S", seq=1000)

# Combine IP and TCP layers
syn_pkt = ip / tcp

# Display packet details
syn_pkt.show()

# Send the packet
send(syn_pkt)
```

(c) TCP Listener (Python)

```
import socket

# Host '0.0.0.0' means listen on all available interfaces
HOST = '0.0.0.0'
PORT = 8080  # Listening port (same as Scapy TCP destination port)

# Create a TCP socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Allow address reuse (helps restart server quickly)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

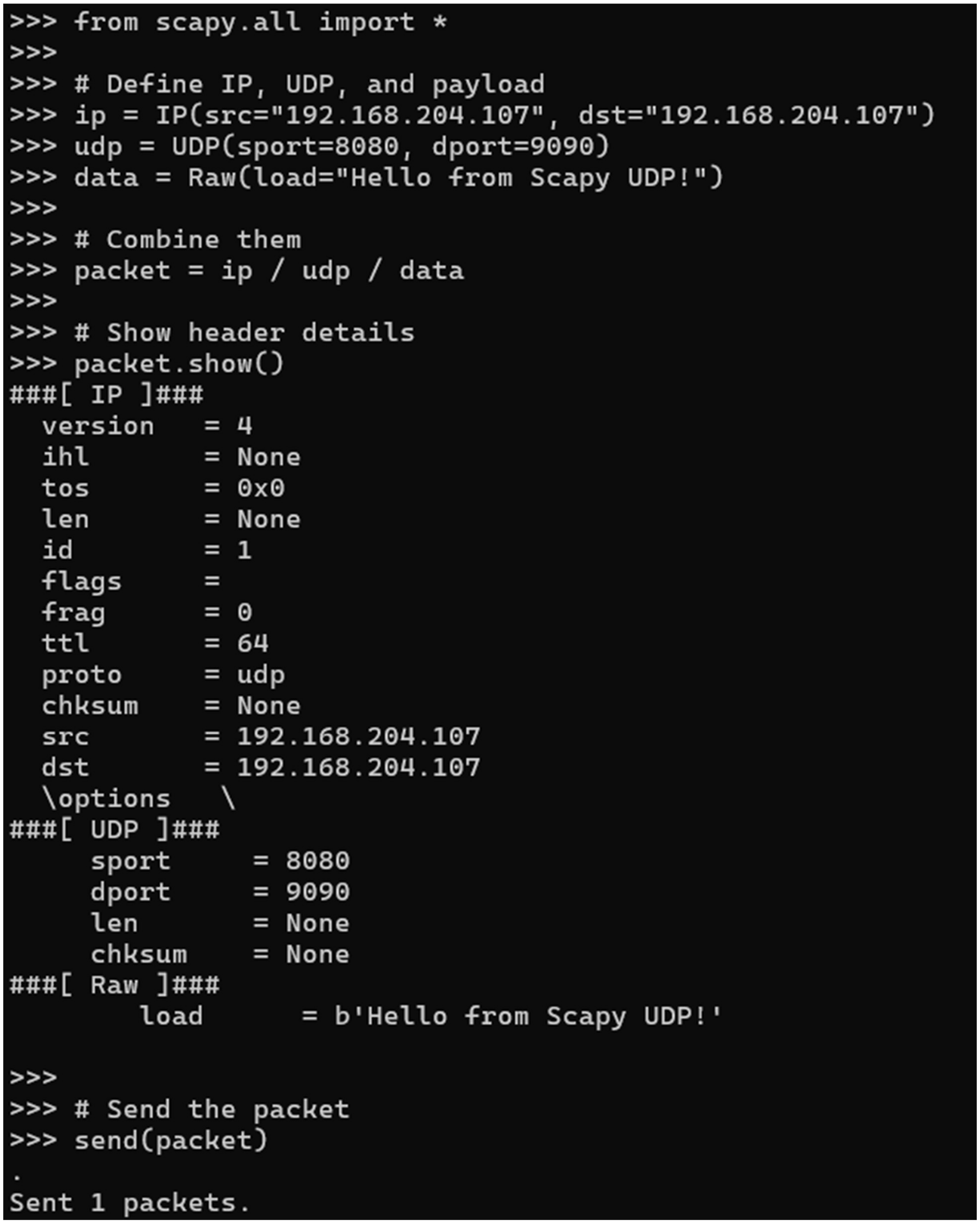
```
# Bind socket to the host and port
s.bind((HOST, PORT))

# Start listening for incoming connections
s.listen(1)
print(f"Listening on {HOST}:{PORT} ...")

# Accept one connection (this blocks until a client connects)
conn, addr = s.accept()
print("Accepted connection from", addr)

# Receive any data sent by the client (up to 1024 bytes)
data = conn.recv(1024)
print("Received:", data)

# Send a simple acknowledgment to the client
conn.sendall(b"Hello from listener")

# Close connection and socket
conn.close()
s.close()
```

Screenshots:



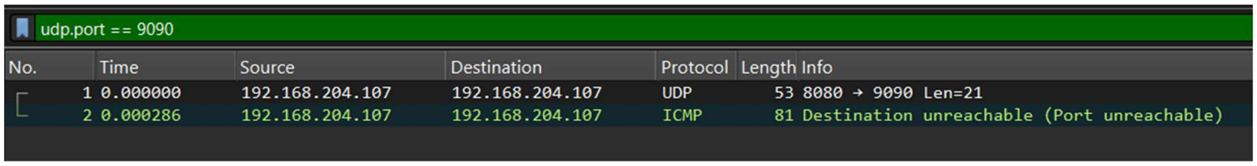
```
C:\Users\psssa>scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: Can't import python-cryptography v1.7+. Disabled PKI & TLS crypto-related features.
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryption/authentication.
WARNING: No alternative Python interpreters found ! Using standard Python shell instead.
INFO: Using the default Python shell: History,Colors are disabled.

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.6.1
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets like I craft my beer.
       scccccp///pSP///p          p//Y   |               -- Jean De Clerck
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
```

Figure 1: Scapy launched successfully in Windows 11 environment.



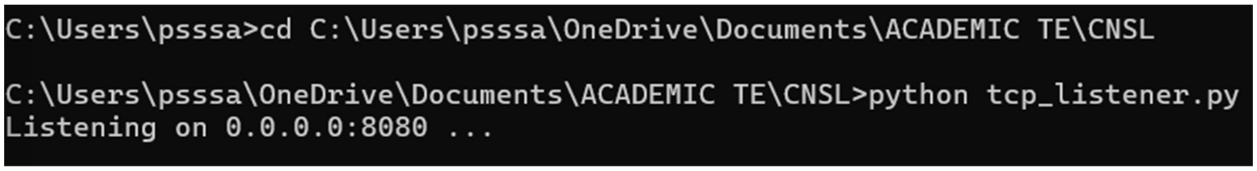
```
>>> from scapy.all import *
>>>
>>> # Define IP, UDP, and payload
>>> ip = IP(src="192.168.204.107", dst="192.168.204.107")
>>> udp = UDP(sport=8080, dport=9090)
>>> data = Raw(load="Hello from Scapy UDP!")
>>>
>>> # Combine them
>>> packet = ip / udp / data
>>>
>>> # Show header details
>>> packet.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 192.168.204.107
  dst       = 192.168.204.107
  \options   \
###[ UDP ]###
     sport     = 8080
     dport     = 9090
     len       = None
     chksum    = None
###[ Raw ]###
        load      = b'Hello from Scapy UDP!'

>>>
>>> # Send the packet
>>> send(packet)
.
Sent 1 packets.
```

Figure 2: UDP packet creation and sending via Scapy.



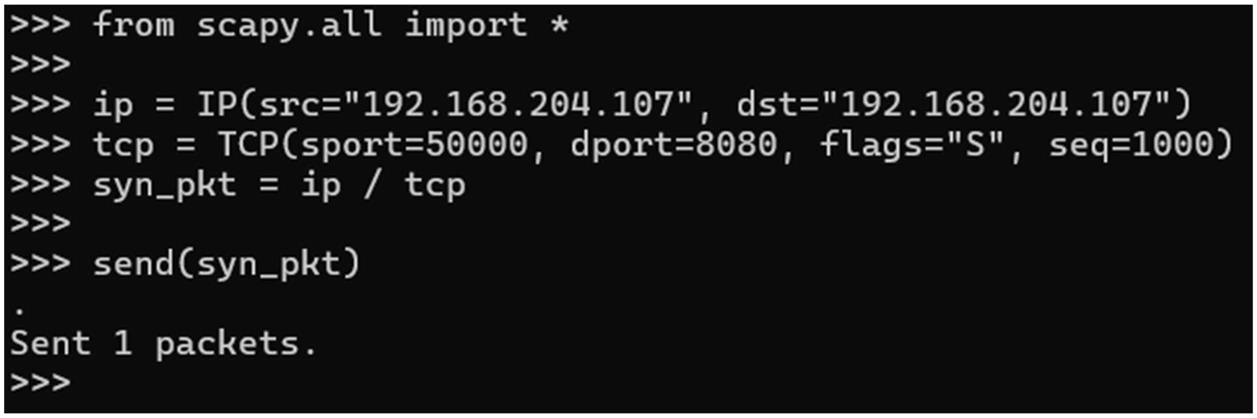
```
udp.port == 9090
No.   Time        Source            Destination       Protocol Length Info
    1 0.000000    192.168.204.107   192.168.204.107   UDP      53 8080 → 9090 Len=21
    2 0.000286    192.168.204.107   192.168.204.107   ICMP     81 Destination unreachable (Port unreachable)
```

Figure 3: Wireshark capture of UDP packet (8080→9090) and ICMP Port Unreachable response.



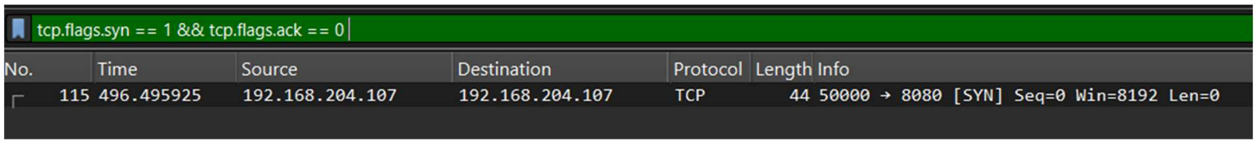
```
C:\Users\psssa>cd C:\Users\psssa\OneDrive\Documents\ACADEMIC TE\CNSL

C:\Users\psssa\OneDrive\Documents\ACADEMIC TE\CNSL>python tcp_listener.py
Listening on 0.0.0.0:8080 ...
```

Figure 4: TCP listener running on port 8080.



```
>>> from scapy.all import *
>>>
>>> ip = IP(src="192.168.204.107", dst="192.168.204.107")
>>> tcp = TCP(sport=50000, dport=8080, flags="S", seq=1000)
>>> syn_pkt = ip / tcp
>>>
>>> send(syn_pkt)
.
Sent 1 packets.
>>>
```

Figure 5: TCP SYN packet creation and sending in Scapy.



```
tcp.flags.syn == 1 && tcp.flags.ack == 0
No.   Time          Source            Destination       Protocol Length Info
  115 496.495925    192.168.204.107   192.168.204.107   TCP      44 50000 → 8080 [SYN] Seq=0 Win=8192 Len=0
```

Figure 6: Wireshark capture of TCP SYN packet (50000→8080) with SYN flag.