Week 5 lab reports

Members:

1. Harshita 191cs120
2. Keerthana 191cs231

1.Develop a code to illustrate a secure socket connection between client and server.

IMPLEMENTATION:
SSL connection between client and server using python.

SERVER --

- Firstly we created a socket and bind it to the address and the port.
- If we want to create a secure connection, we must create a SSL default context for the server.
- Now we have to load the key and all the required certificates to create the function.
- We created the server certificate using the following command:

```
openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout serb=ver.key -out server.crtr
```

  certificate and key files will be downloaded in the given path.

- We used the

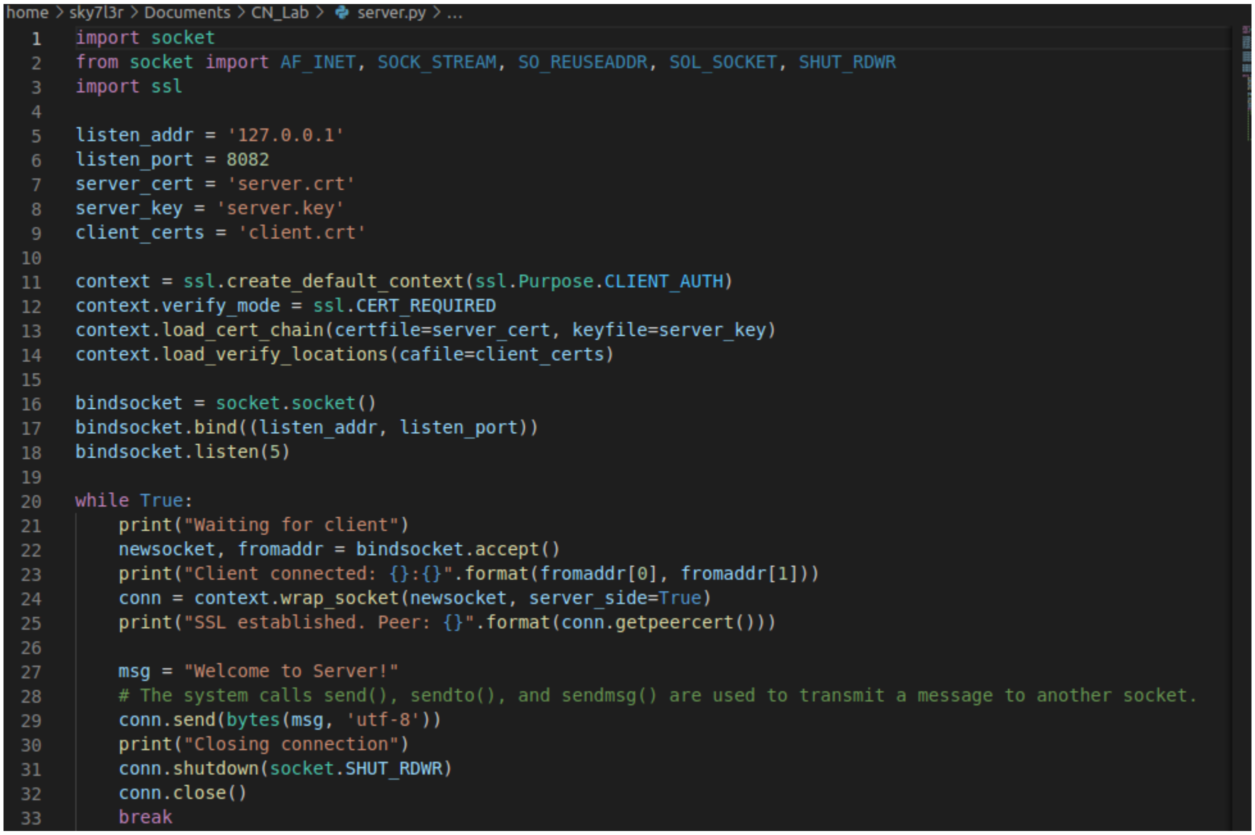
  load_cert_chain() function to load the certificate and key into the program file.

- Now we created a function that waits for a client and accepts when the address matches and then uses ssl.wrap_socket() making server_side=True to enable client certificates.
- Then we established the SSL connection using getpeercert() function.
- Finally we are free to send the messages from the server to the client and vice versa.

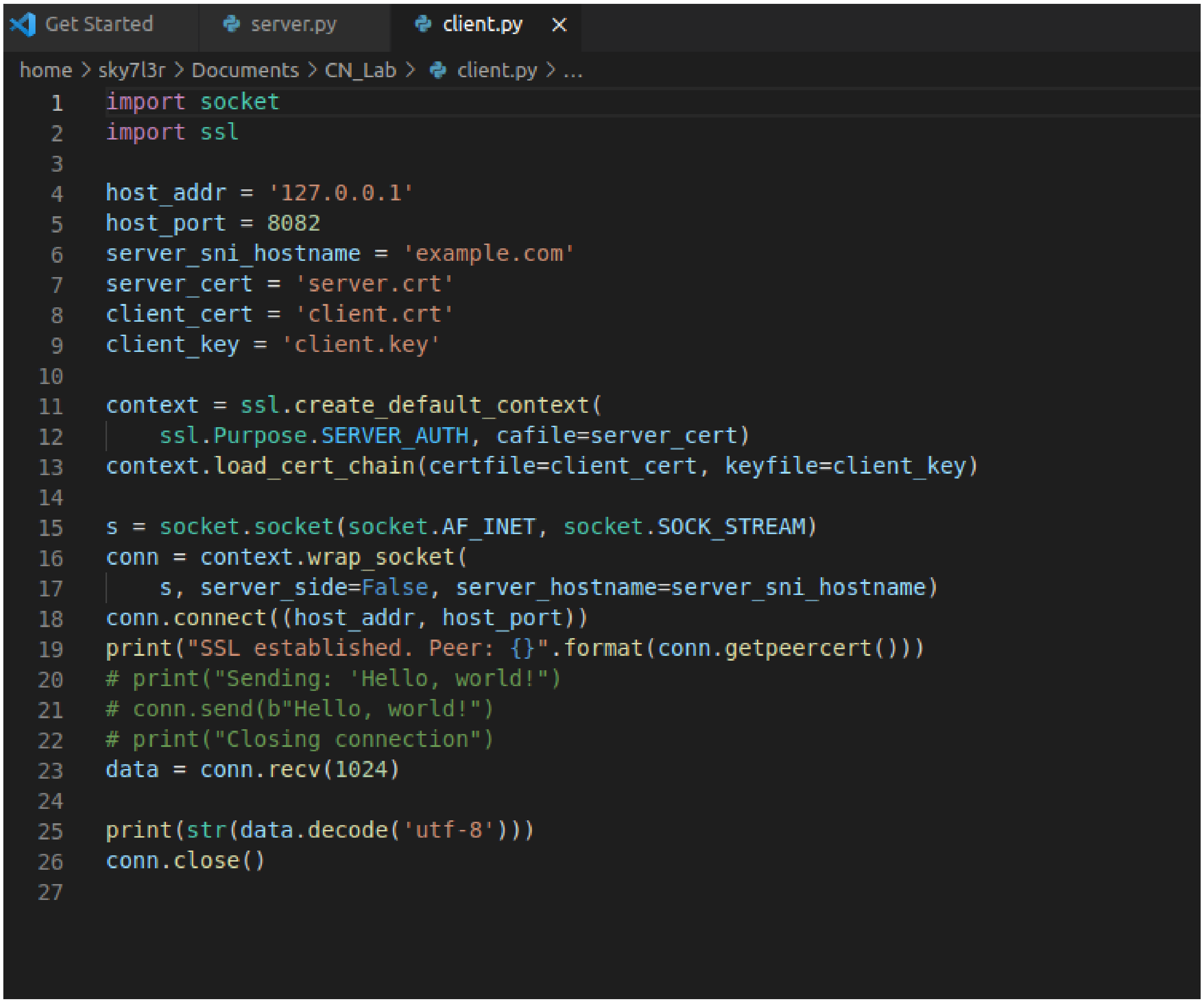
  CLIENT--

- We did the same thing here too. We have created a socket and created the default SSL context in order to establish a secure connection.
- The client certificate can be created in a similar way to the server.
- Use the load_cert_chain() function to load the certificate and key.

- Similarly, use ssl.wrap_socket() to enable the certificates and establish the SSL connection using getpeercert() function.
- Now, we can send and receive messages from the server.



```python
import socket
from socket import AF_INET, SOCK_STREAM, SO_REUSEADDR, SOL_SOCKET, SHUT_RDWR
import ssl

listen_addr = '127.0.0.1'
listen_port = 8082
server_cert = 'server.crt'
server_key = 'server.key'
client_certs = 'client.crt'

context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.verify_mode = ssl.CERT_REQUIRED
context.load_cert_chain(certfile=server_cert, keyfile=server_key)
context.load_verify_locations(cafile=client_certs)

bindsocket = socket.socket()
bindsocket.bind((listen_addr, listen_port))
bindsocket.listen(5)

while True:
    print("Waiting for client")
    newsocket, fromaddr = bindsocket.accept()
    print("Client connected: {}:{}".format(fromaddr[0], fromaddr[1]))
    conn = context.wrap_socket(newsocket, server_side=True)
    print("SSL established. Peer: {}".format(conn.getpeercert()))

    msg = "Welcome to Server!"
    # The system calls send(), sendto(), and sendmsg() are used to transmit a message to another socket.
    conn.send(bytes(msg, 'utf-8'))
    print("Closing connection")
    conn.shutdown(socket.SHUT_RDWR)
    conn.close()
    break
```



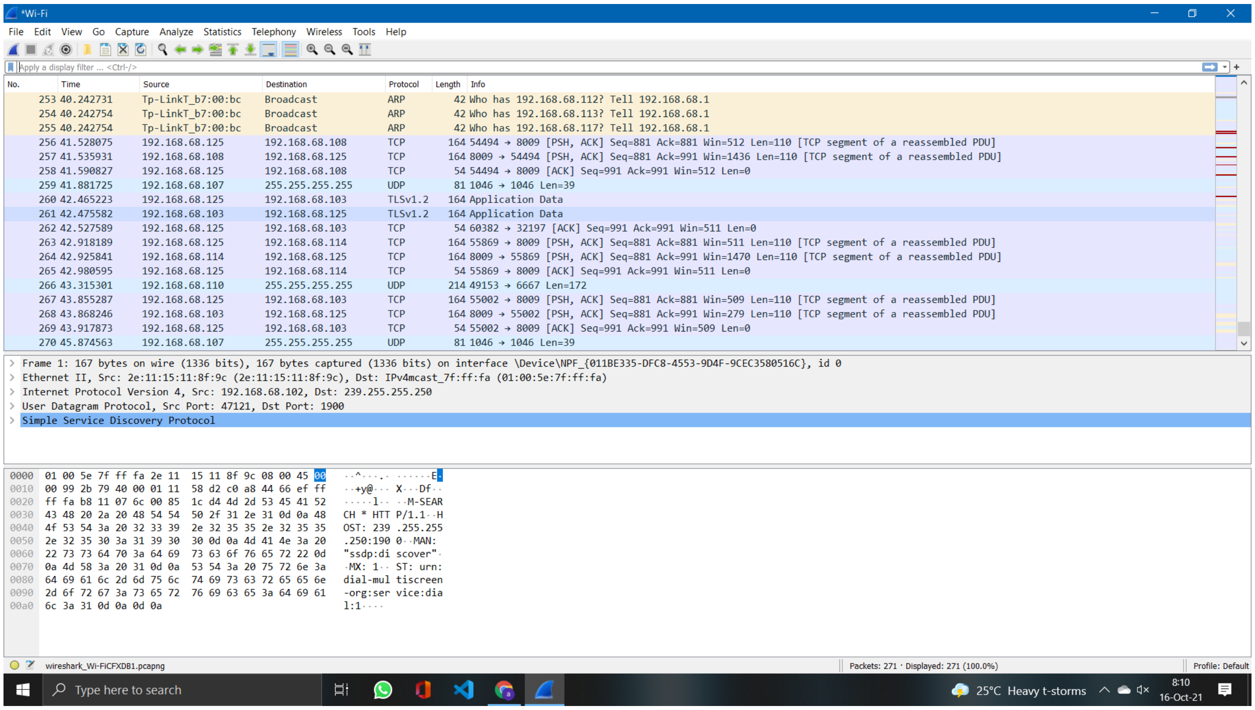
```python
import socket
import ssl

host_addr = '127.0.0.1'
host_port = 8082
server_sni_hostname = 'example.com'
server_cert = 'server.crt'
client_cert = 'client.crt'
client_key = 'client.key'

context = ssl.create_default_context(
    ssl.Purpose.SERVER_AUTH, cafile=server_cert)
context.load_cert_chain(certfile=client_cert, keyfile=client_key)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn = context.wrap_socket(
    s, server_side=False, server_hostname=server_sni_hostname)
conn.connect((host_addr, host_port))
print("SSL established. Peer: {}".format(conn.getpeercert()))
# print("Sending: 'Hello, world!")
# conn.send(b"Hello, world!")
# print("Closing connection")
data = conn.recv(1024)

print(str(data.decode('utf-8')))
conn.close()

```

2. Capture TCP Packets and:

a. Analyse the three-way handshake during the establishment of the communication.

b. Identify if there are any retransmitted segments.
IMPLEMENTATION:

- Firstly Open Wireshark and choose a local network to capture network traffic.
- Clear the present traffic and open a browser and access a website.
- Now you can see network traffic captured in wireshark.
- Filter the TCP packets using the tcp filter.
- Now we can see that the top three packets of the traffic will be [SYN],[SYN, ACK] and [ACK] respectively. This implies that the three way hand shake is done and the client can make a request to server.
- Now we must analyse the above three identified packets.
- SYN: We can see that the seq : 0 and length: 0 and in the flags, the synchronization number is set to 1 and the acknowledgement number is set to 0 as it is the first request going to the server.
- SYN,ACK : Here we can see that the seq: 0 and Ack: 1 which means that the request is gone to the server. The synchronization and acknowledgement flags are both set to 1.
- ACK: Here we can see that the seq: 1 and ack: 1 and in the flags, the synchronization is set to 0 and acknowledgement flag is set to 1 which means that the connection is established.
- Since, a successful three way handshake is done, the real data communication can be started.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Type here to search
25°C Heavy t-storms
8:10
16-Oct-21

```
No.   Time        Source             Destination       Protocol  Length Info
  253 40.242731   Tp-LinkT_b7:00:bc  Broadcast         ARP          42 Who has 192.168.68.112? Tell 192.168.68.1
  254 40.242754   Tp-LinkT_b7:00:bc  Broadcast         ARP          42 Who has 192.168.68.113? Tell 192.168.68.1
  255 40.242754   Tp-LinkT_b7:00:bc  Broadcast         ARP          42 Who has 192.168.68.117? Tell 192.168.68.1
  256 41.528075   192.168.68.125     192.168.68.108    TCP         164 54494 → 8009 [PSH, ACK] Seq=881 Ack=881 Win=512 Len=110 [TCP segment of a reassembled PDU]
  257 41.535931   192.168.68.108     192.168.68.125    TCP         164 8009 → 54494 [PSH, ACK] Seq=881 Ack=991 Win=1436 Len=110 [TCP segment of a reassembled PDU]
  258 41.590827   192.168.68.125     192.168.68.108    TCP          54 54494 → 8009 [ACK] Seq=991 Ack=991 Win=512 Len=0
  259 41.881725   192.168.68.107     255.255.255.255   UDP          81 1046 → 1046 Len=39
  260 42.465223   192.168.68.125     192.168.68.103    TLSv1.2     164 Application Data
  261 42.475582   192.168.68.103     192.168.68.125    TLSv1.2     164 Application Data
  262 42.527589   192.168.68.125     192.168.68.103    TCP          54 60382 → 32197 [ACK] Seq=991 Ack=991 Win=511 Len=0
  263 42.918189   192.168.68.125     192.168.68.114    TCP         164 55869 → 8009 [PSH, ACK] Seq=881 Ack=881 Win=511 Len=110 [TCP segment of a reassembled PDU]
  264 42.925841   192.168.68.114     192.168.68.125    TCP         164 8009 → 55869 [PSH, ACK] Seq=881 Ack=991 Win=1470 Len=110 [TCP segment of a reassembled PDU]
  265 42.980595   192.168.68.125     192.168.68.114    TCP          54 55869 → 8009 [ACK] Seq=991 Ack=991 Win=511 Len=0
  266 43.315301   192.168.68.110     255.255.255.255   UDP         214 49153 → 6667 Len=172
  267 43.855287   192.168.68.125     192.168.68.103    TCP         164 55002 → 8009 [PSH, ACK] Seq=881 Ack=881 Win=509 Len=110 [TCP segment of a reassembled PDU]
  268 43.868246   192.168.68.103     192.168.68.125    TCP         164 8009 → 55002 [PSH, ACK] Seq=881 Ack=991 Win=279 Len=110 [TCP segment of a reassembled PDU]
  269 43.917873   192.168.68.125     192.168.68.103    TCP          54 55002 → 8009 [ACK] Seq=991 Ack=991 Win=509 Len=0
  270 45.874563   192.168.68.107     255.255.255.255   UDP          81 1046 → 1046 Len=39
```

```
> Frame 1: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: 2e:11:15:11:8f:9c (2e:11:15:11:8f:9c), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
> Internet Protocol Version 4, Src: 192.168.68.102, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 47121, Dst Port: 1900
> Simple Service Discovery Protocol
```

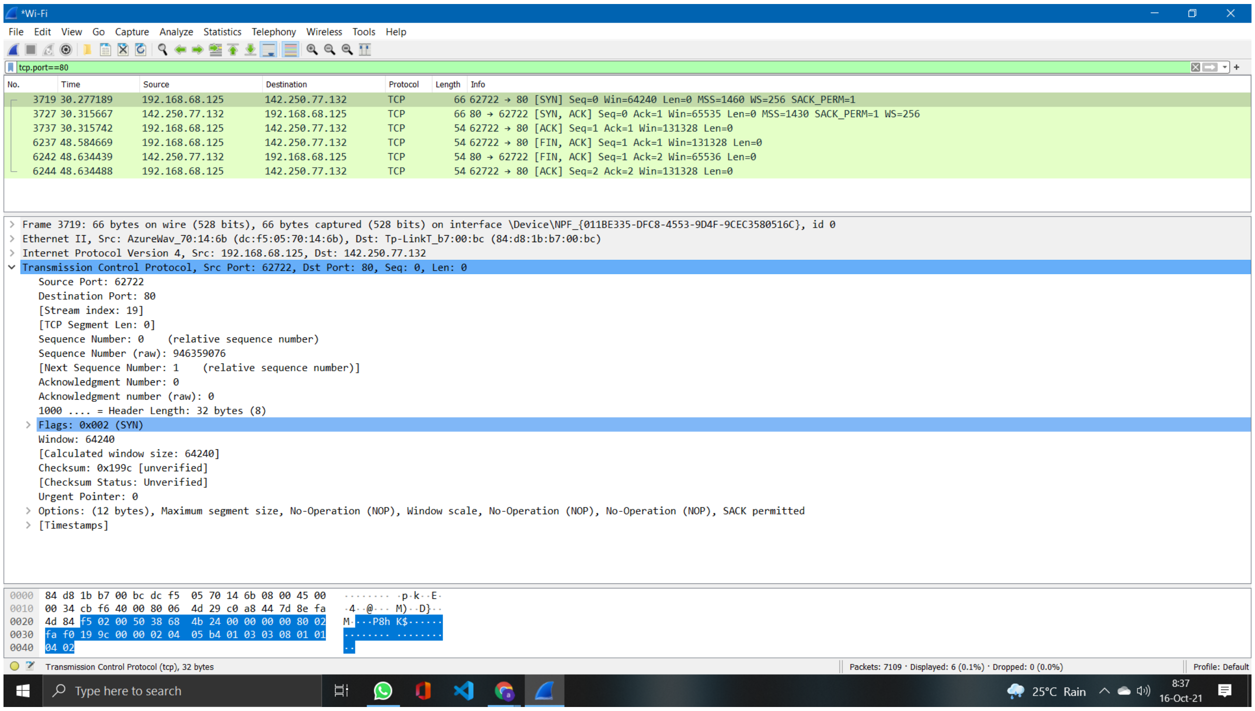
```
0000  01 00 5e 7f ff fa 2e 11  15 11 8f 9c 08 00 45 00   ··^····· ······E·
0010  00 99 2b 79 40 00 01 11  58 d2 c0 a8 44 66 ef ff   ··+y@··· X···Df··
0020  ff fa b8 11 07 6c 00 85  1c d4 4d 2d 53 45 41 52   ·····l·· ··M-SEAR
0030  43 48 20 2a 20 48 54 54  50 2f 31 2e 31 0d 0a 48   CH * HTT P/1.1··H
0040  4f 53 54 3a 20 32 33 39  2e 32 35 35 2e 32 35 35   OST: 239 .255.255
0050  2e 32 35 30 3a 31 39 30  30 0d 0a 4d 41 4e 3a 20   .250:190 0··MAN:
0060  22 73 73 64 70 3a 64 69  73 63 6f 76 65 72 22 0d   "ssdp:di scover"·
0070  0a 4d 58 3a 20 31 0d 0a  53 54 3a 20 75 72 6e 3a   ·MX: 1·· ST: urn:
0080  64 69 61 6c 2d 6d 75 6c  74 69 73 63 72 65 65 6e   dial-mul tiscreen
0090  2d 6f 72 67 3a 73 65 72  76 69 63 65 3a 64 69 61   -org:ser vice:dia
00a0  6c 3a 31 0d 0a 0d 0a                               l:1····
```

wireshark_Wi-FiCFXDB1.pcapng | Packets: 271 · Displayed: 271 (100.0%) | Profile: Default

Observe the packet details in the middle Wireshark packet details pane. Notice that it is an Ethernet II / Internet Protocol Version 4 / Transmission Control Protocol frame

Then

Capturing TCP packets.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Type here to search
25°C Rain
8:37
16-Oct-21

tcp.port==80

```
No.   Time        Source             Destination       Protocol  Length Info
 3719 30.277189   192.168.68.125     142.250.77.132    TCP          66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 3727 30.315667   142.250.77.132     192.168.68.125    TCP          66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
 3737 30.315742   192.168.68.125     142.250.77.132    TCP          54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 6237 48.584669   192.168.68.125     142.250.77.132    TCP          54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
 6242 48.634439   142.250.77.132     192.168.68.125    TCP          54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
 6244 48.634488   192.168.68.125     142.250.77.132    TCP          54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

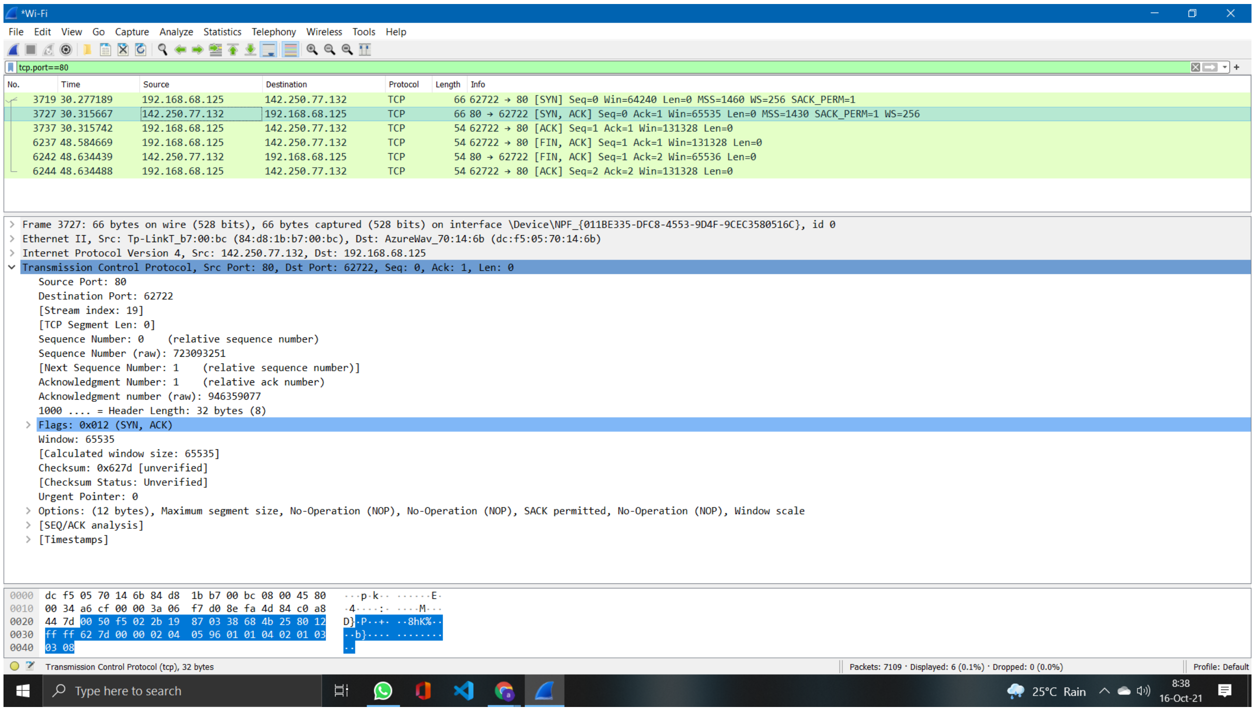
```
> Frame 3719: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: AzureWav_70:14:6b (dc:f5:05:70:14:6b), Dst: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc)
> Internet Protocol Version 4, Src: 192.168.68.125, Dst: 142.250.77.132
v Transmission Control Protocol, Src Port: 62722, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 62722
    Destination Port: 80
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 946359076
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x199c [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [Timestamps]
```

```
0000  84 d8 1b b7 00 bc dc f5  05 70 14 6b 08 00 45 00   ········ ·p·k··E·
0010  00 34 cb f6 40 00 80 06  4d 29 c0 a8 44 7d 8e fa   ·4··@··· M)··D}··
0020  4d 84 f5 02 00 50 38 68  4b 24 00 00 00 00 80 02   M····P8h K$······
0030  fa f0 19 9c 00 00 02 04  05 b4 01 03 03 08 01 01   ········ ········
0040  04 02                                              ··
```

Transmission Control Protocol (tcp), 32 bytes | Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%) | Profile: Default

Observe the flag settings. Notice that SYN is set, indicating the first segment in the TCP three-way handshake.

Observe the Sequence number. Notice that it is 0 (relative sequence number). To see the actual sequence number, select the Sequence number to highlight the sequence number in the bottom Wireshark bytes pane



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==80

```
No.   Time      Source          Destination     Protocol Length Info
 3719 30.277189 192.168.68.125  142.250.77.132  TCP        66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 3727 30.315667 142.250.77.132  192.168.68.125  TCP        66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
 3737 30.315742 192.168.68.125  142.250.77.132  TCP        54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 6237 48.584669 192.168.68.125  142.250.77.132  TCP        54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
 6242 48.634439 142.250.77.132  192.168.68.125  TCP        54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
 6244 48.634488 192.168.68.125  142.250.77.132  TCP        54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

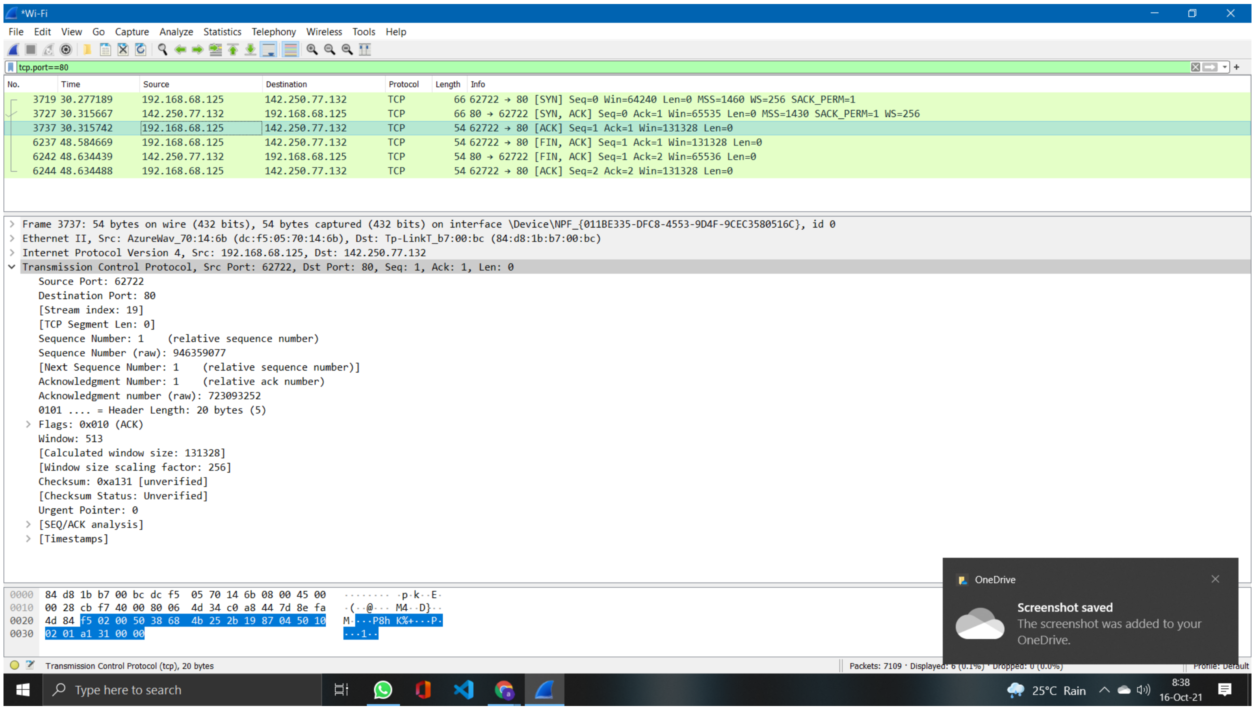
```
> Frame 3727: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc), Dst: AzureWav_70:14:6b (dc:f5:05:70:14:6b)
> Internet Protocol Version 4, Src: 142.250.77.132, Dst: 192.168.68.125
v Transmission Control Protocol, Src Port: 80, Dst Port: 62722, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 62722
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 723093251
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 946359077
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x627d [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  > [SEQ/ACK analysis]
  > [Timestamps]
```

```
0000  dc f5 05 70 14 6b 84 d8  1b b7 00 bc 08 00 45 80   ···p·k·· ······E·
0010  00 34 a6 cf 00 00 3a 06  f7 d0 8e fa 4d 84 c0 a8   ·4····:· ····M···
0020  44 7d 00 50 f5 02 2b 19  87 03 38 68 4b 25 80 12   D}·P··+· ··8hK%··
0030  ff ff 62 7d 00 00 02 04  05 96 01 01 04 02 01 03   ··b}···· ········
0040  03 08                                              ··
```

```
Transmission Control Protocol (tcp), 32 bytes    Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%)    Profile: Default
```

Observe the Acknowledgement number. Notice that it is 1 (relative ack number). To see the actual acknowledgement number, select Acknowledgement number to highlight the acknowledgement number in the bottom pane. Notice that the actual acknowledgement number is one greater than the sequence number in the previous segment.

Observe the flag settings. Notice that SYN and ACK are set, indicating the second segment in the TCP three-way handshake.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
25°C Rain
8:38
16-Oct-21

tcp.port==80

```
No.   Time       Source          Destination     Protocol Length Info
 3719 30.277189  192.168.68.125  142.250.77.132  TCP          66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 3727 30.315667  142.250.77.132  192.168.68.125  TCP          66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
 3737 30.315742  192.168.68.125  142.250.77.132  TCP          54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 6237 48.584669  192.168.68.125  142.250.77.132  TCP          54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
 6242 48.634439  142.250.77.132  192.168.68.125  TCP          54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
 6244 48.634488  192.168.68.125  142.250.77.132  TCP          54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

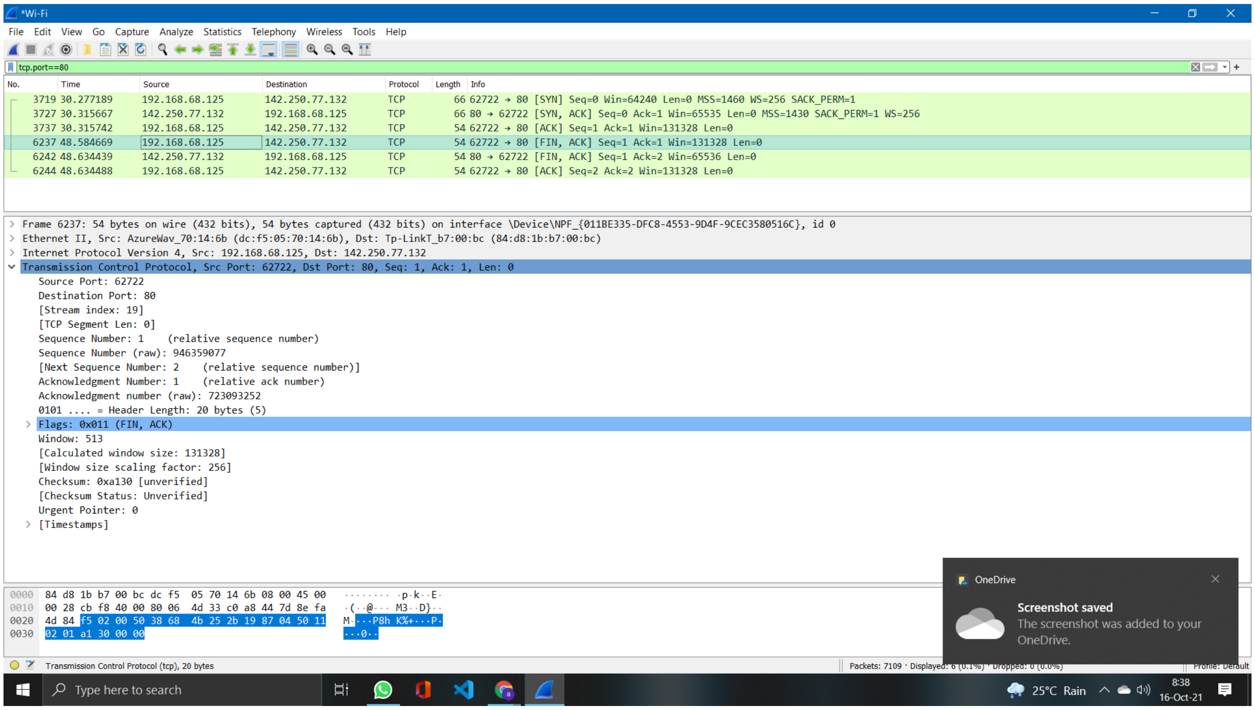
```
> Frame 3737: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: AzureWav_70:14:6b (dc:f5:05:70:14:6b), Dst: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc)
> Internet Protocol Version 4, Src: 192.168.68.125, Dst: 142.250.77.132
v Transmission Control Protocol, Src Port: 62722, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 62722
    Destination Port: 80
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 946359077
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 723093252
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
    Checksum: 0xa131 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

```
0000  84 d8 1b b7 00 bc dc f5  05 70 14 6b 08 00 45 00   ········ ·p·k··E·
0010  00 28 cb f7 40 00 80 06  4d 34 c0 a8 44 7d 8e fa   ·(··@··· M4··D}··
0020  4d 84 f5 02 00 50 38 68  4b 25 2b 19 87 04 50 10   M····P8h K%+···P·
0030  02 01 a1 31 00 00                                  ···1··
```

Transmission Control Protocol (tcp), 20 bytes | Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%) | Profile: Default

Observe the Acknowledgement number. Notice that it is 1 (relative ack number). To see the actual acknowledgement number, select Acknowledgement number to highlight the acknowledgement number in the bottom pane.

Observe the flag settings. Notice that ACK is set, indicating the third segment in the TCP three-way handshake. The client has established a TCP connection with the server


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==80
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
25°C Rain
8:38
16-Oct-21

```
No.     Time         Source           Destination      Protocol  Length  Info
   3719 30.277189    192.168.68.125   142.250.77.132   TCP       66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   3727 30.315667    142.250.77.132   192.168.68.125   TCP       66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
   3737 30.315742    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
   6237 48.584669    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
   6242 48.634439    142.250.77.132   192.168.68.125   TCP       54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
   6244 48.634488    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

```
> Frame 6237: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: AzureWav_70:14:6b (dc:f5:05:70:14:6b), Dst: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc)
> Internet Protocol Version 4, Src: 192.168.68.125, Dst: 142.250.77.132
v Transmission Control Protocol, Src Port: 62722, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
     Source Port: 62722
     Destination Port: 80
     [Stream index: 19]
     [TCP Segment Len: 0]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 946359077
     [Next Sequence Number: 2    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 723093252
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x011 (FIN, ACK)
     Window: 513
     [Calculated window size: 131328]
     [Window size scaling factor: 256]
     Checksum: 0xa130 [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   > [Timestamps]
```

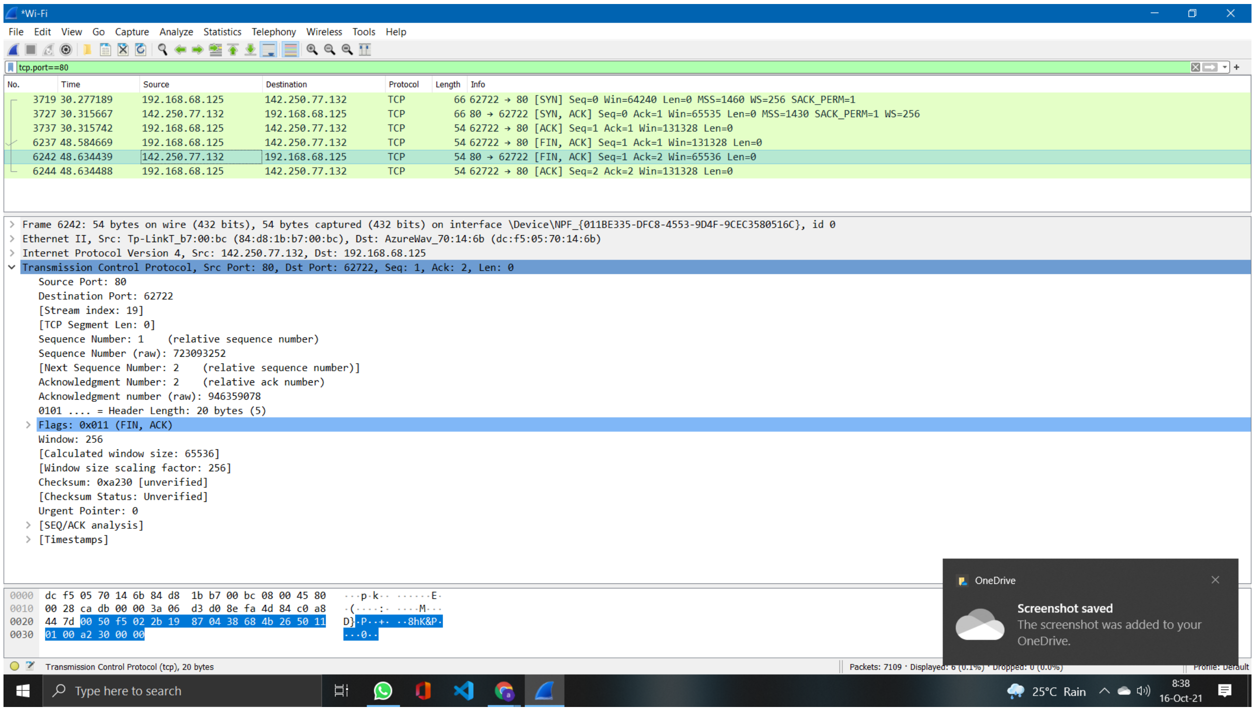
```
0000  84 d8 1b b7 00 bc dc f5  05 70 14 6b 08 00 45 00   ········ ·p·k··E·
0010  00 28 cb f8 40 00 80 06  4d 33 c0 a8 44 7d 8e fa   ·(··@··· M3··D}··
0020  4d 84 f5 02 00 50 38 68  4b 25 2b 19 87 04 50 11   M····P8h K%+···P·
0030  02 01 a1 30 00 00                                  ···0··
```

Transmission Control Protocol (tcp), 20 bytes

Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%)

Profile: Default

Observe the flag settings. Notice that FIN and ACK are set, indicating the first segment in the TCP teardown handshake. The client has indicated it is closing the TCP connection with the server.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==80
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
25°C Rain
8:38
16-Oct-21

```
No.     Time         Source           Destination      Protocol  Length  Info
   3719 30.277189    192.168.68.125   142.250.77.132   TCP       66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   3727 30.315667    142.250.77.132   192.168.68.125   TCP       66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
   3737 30.315742    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
   6237 48.584669    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
   6242 48.634439    142.250.77.132   192.168.68.125   TCP       54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
   6244 48.634488    192.168.68.125   142.250.77.132   TCP       54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

```
> Frame 6242: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc), Dst: AzureWav_70:14:6b (dc:f5:05:70:14:6b)
> Internet Protocol Version 4, Src: 142.250.77.132, Dst: 192.168.68.125
v Transmission Control Protocol, Src Port: 80, Dst Port: 62722, Seq: 1, Ack: 2, Len: 0
     Source Port: 80
     Destination Port: 62722
     [Stream index: 19]
     [TCP Segment Len: 0]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 723093252
     [Next Sequence Number: 2    (relative sequence number)]
     Acknowledgment Number: 2    (relative ack number)
     Acknowledgment number (raw): 946359078
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x011 (FIN, ACK)
     Window: 256
     [Calculated window size: 65536]
     [Window size scaling factor: 256]
     Checksum: 0xa230 [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   > [SEQ/ACK analysis]
   > [Timestamps]
```

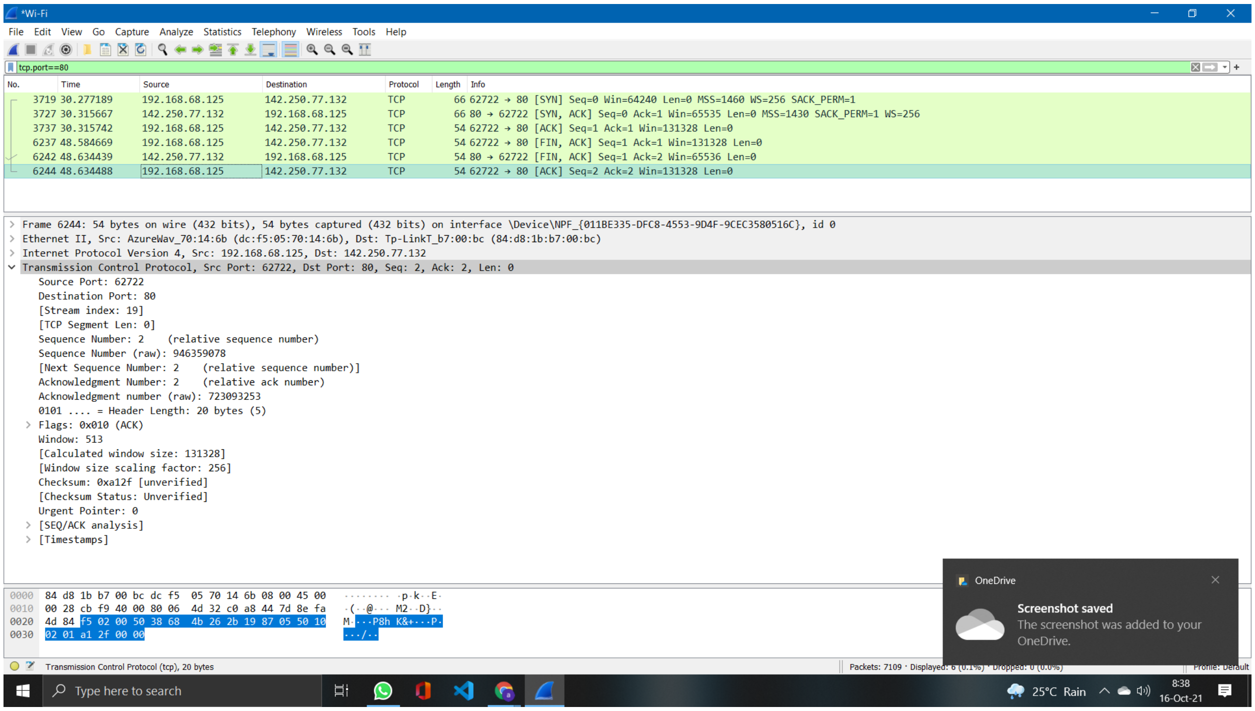
```
0000  dc f5 05 70 14 6b 84 d8  1b b7 00 bc 08 00 45 80   ···p·k·· ······E·
0010  00 28 ca db 00 00 3a 06  d3 d0 8e fa 4d 84 c0 a8   ·(····:· ····M···
0020  44 7d 00 50 f5 02 2b 19  87 04 38 68 4b 26 50 11   D}·P··+· ··8hK&P·
0030  01 00 a2 30 00 00                                  ···0··
```

Transmission Control Protocol (tcp), 20 bytes

Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%)

Profile: Default

Observe the flag settings. Notice that FIN and ACK are set, indicating the second segment in the TCP three-way handshake. The server has indicated it is closing the TCP connection with the client.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==80

```
No.   Time       Source          Destination     Protocol Length Info
 3719 30.277189  192.168.68.125  142.250.77.132  TCP      66 62722 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 3727 30.315667  142.250.77.132  192.168.68.125  TCP      66 80 → 62722 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
 3737 30.315742  192.168.68.125  142.250.77.132  TCP      54 62722 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 6237 48.584669  192.168.68.125  142.250.77.132  TCP      54 62722 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
 6242 48.634439  142.250.77.132  192.168.68.125  TCP      54 80 → 62722 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0
 6244 48.634488  192.168.68.125  142.250.77.132  TCP      54 62722 → 80 [ACK] Seq=2 Ack=2 Win=131328 Len=0
```

```
> Frame 6244: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{011BE335-DFC8-4553-9D4F-9CEC3580516C}, id 0
> Ethernet II, Src: AzureWav_70:14:6b (dc:f5:05:70:14:6b), Dst: Tp-LinkT_b7:00:bc (84:d8:1b:b7:00:bc)
> Internet Protocol Version 4, Src: 192.168.68.125, Dst: 142.250.77.132
v Transmission Control Protocol, Src Port: 62722, Dst Port: 80, Seq: 2, Ack: 2, Len: 0
     Source Port: 62722
     Destination Port: 80
     [Stream index: 19]
     [TCP Segment Len: 0]
     Sequence Number: 2    (relative sequence number)
     Sequence Number (raw): 946359078
     [Next Sequence Number: 2    (relative sequence number)]
     Acknowledgment Number: 2    (relative ack number)
     Acknowledgment number (raw): 723093253
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x010 (ACK)
     Window: 513
     [Calculated window size: 131328]
     [Window size scaling factor: 256]
     Checksum: 0xa12f [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   > [SEQ/ACK analysis]
   > [Timestamps]
```

```
0000  84 d8 1b b7 00 bc dc f5  05 70 14 6b 08 00 45 00   ········ ·p·k··E·
0010  00 28 cb f9 40 00 80 06  4d 32 c0 a8 44 7d 8e fa   ·(··@··· M2··D}··
0020  4d 84 f5 02 00 50 38 68  4b 26 2b 19 87 05 50 10   M····P8h K&+···P·
0030  02 01 a1 2f 00 00                                  ···/··
```

Transmission Control Protocol (tcp), 20 bytes | Packets: 7109 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%) | Profile: Default

OneDrive

Screenshot saved
The screenshot was added to your OneDrive.

Close Wireshark to complete this activity. **Quit without Saving** to discard the captured traffic

B) Yes, there can be retransmitted segments, when the server is trying to establish a connection but doesn't receive the response within a stipulated time

=