

PES1201800410

sem V

roll no 24

Prashanth A R

Task 1

UDP Server

The screenshot shows a terminal window titled '/bin/bash' with the command 'PES1201800410@10.0.7.24-server\$ python UDPServer.py' entered. The output 'The server is ready to receive' is displayed. The terminal window has a blue header bar and a green footer bar showing the date and time: "VM" 12:49 20-Oct-20.

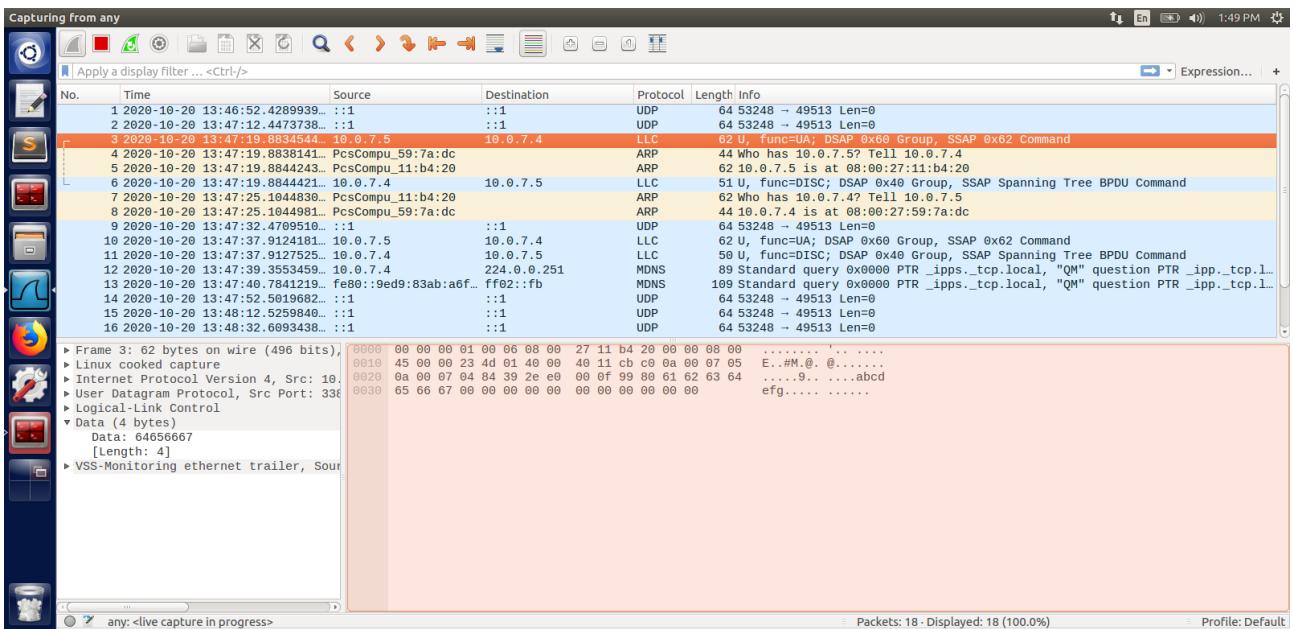
```
/bin/bash
PES1201800410@10.0.7.24-server$ python UDPServer.py
The server is ready to receive
[0] 0:python*
```

UDP Client

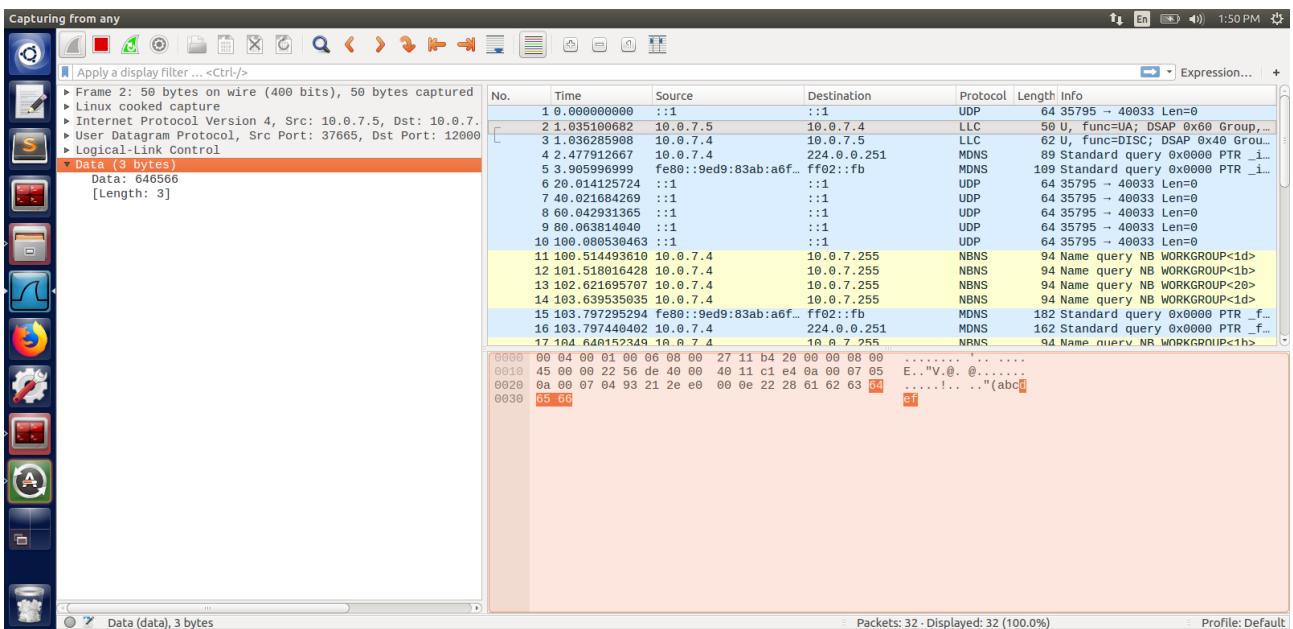
The screenshot shows a terminal window titled '/bin/bash' with the command 'PES1201800410@10.0.7.24-client\$ python UDPClient.py' entered. The input 'Input lowercase sentence: abcdef' is shown, followed by the output 'ABCDEF'. The terminal window has a blue header bar and a green footer bar showing the date and time: "VM" 12:51 20-Oct-20.

```
/bin/bash
PES1201800410@10.0.7.24-client$ python UDPClient.py
Input lowercase sentence: abcdef
ABCDEF
PES1201800410@10.0.7.24-client:$
[0] 0:python*
```

wireshark udp server output



wireshark udp client output



TCP Server

```
/bin/bash          /bin/bash 139x33
PES1201800410@10.0.7.24-server$ python TCPServer.py
The server is ready to receive
[0] 0:bash*      "VM" 12:52 20-Oct-20
```

TCP Client

```
/bin/bash          /bin/bash 139x33
PES1201800410@10.0.7.24-client$ python TCPClinet.py
Input lowercase sentence:abcdef
From Server: ABCDEF
PES1201800410@10.0.7.24-client:$ [0] 0:python*      "VM" 12:52 20-Oct-20
```

Wireshark TCP server output

Capturing from any

Apply a display Filter ... <Ctrl-/>

No. Time Source Destination Protocol Length Info

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-10-20 13:51:52.7194283...::1		::1	UDP	64	53248 - 49513 Len=0
2	2020-10-20 13:52:07.0819436...10.0.7.4		10.0.7.4	TCP	76	51416 - 12800 [SYN] Seq=3315612986 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TS...
3	2020-10-20 13:52:08.0819744...10.0.7.4		10.0.7.4	TCP	76	12000 - 51416 [SYN, ACK] Seq=3315612987 Ack=3077013413 Win=28906 Len=0 MSS...
4	2020-10-20 13:52:09.06.0823887...10.0.7.5		10.0.7.4	TCP	68	51416 - 12000 [ACK] Seq=3315612987 Ack=3077013413 Win=29312 Len=0 TSval=26...
5	2020-10-20 13:52:09.1718924...10.0.7.5		10.0.7.4	TCP	74	51416 - 12000 [PSH, ACK] Seq=3315612987 Ack=3077013413 Win=29312 Len=6 TSv...
6	2020-10-20 13:52:09.1719712...10.0.7.4		10.0.7.5	TCP	68	12000 - 51416 [ACK] Seq=3077013413 Ack=3315612993 Win=29056 Len=0 TSval=25...
7	2020-10-20 13:52:09.1720492...10.0.7.4		10.0.7.5	TCP	74	12000 - 51416 [PSH, ACK] Seq=3077013413 Ack=3315612993 Win=29056 Len=6 TSv...
8	2020-10-20 13:52:09.1720754...10.0.7.4		10.0.7.5	TCP	68	12000 - 51416 [FIN, ACK] Seq=3077013413 Ack=3315612993 Win=29056 Len=0 TSv...
9	2020-10-20 13:52:09.1725403...10.0.7.5		10.0.7.4	TCP	68	51416 - 12000 [ACK] Seq=3315612993 Ack=3077013419 Win=29312 Len=0 TSval=26...
10	2020-10-20 13:52:09.1725550...10.0.7.5		10.0.7.4	TCP	68	51416 - 12000 [FIN, ACK] Seq=3315612993 Ack=3077013420 Win=29312 Len=0 TSv...
11	2020-10-20 13:52:09.1725634...10.0.7.4		10.0.7.5	TCP	68	12000 - 51416 [ACK] Seq=3077013420 Ack=3315612994 Win=29056 Len=0 TSval=25...
12	2020-10-20 13:52:12.7354332...::1		::1	UDP	64	53248 - 49513 Len=0
13	2020-10-20 13:52:32.7379529...::1		::1	UDP	64	53248 - 49513 Len=0

Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.7.4 (10.0.7.4), Dst: 10.0.7.5 (10.0.7.5)
Transmission Control Protocol, Src Port: 51416 (51416), Dst Port: 12345 (12345)
Data (6 bytes)
Data: 616263646566 [Length: 6]

Packets: 13 - Displayed: 13 (100.0%) Profile: Default

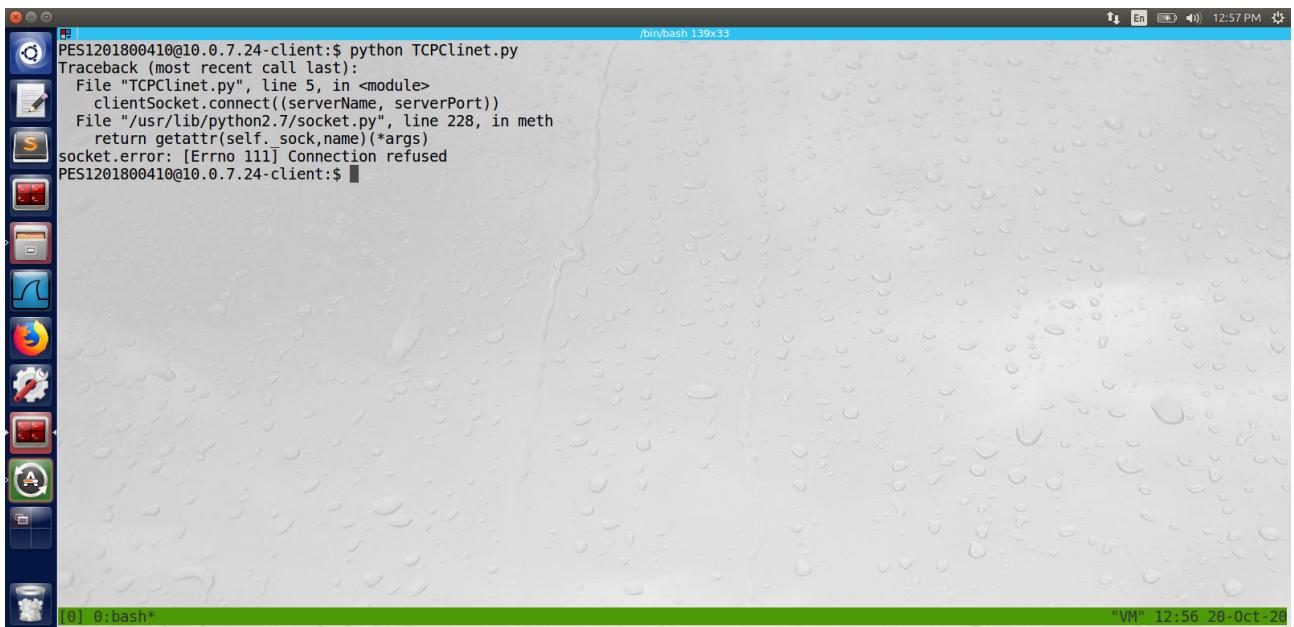
Wireshark TCP client output

The screenshot shows the Wireshark interface with the following details:

- Capturing from any**: The source of the capture is set to "any".
- Apply a display Filter ... <Ctrl-/>**: A display filter is applied.
- Frame 4: 74 bytes on wire (592 bits), 74 bytes captured**: Frame details for frame 4.
- Linux cooked capture**: The capture type is "Linux cooked capture".
- Internet Protocol Version 4, Src: 10.0.7.5, Dst: 10.0.7.4**: IP layer information.
- Transmission Control Protocol, Src Port: 51416, Dst Port**: TCP layer information.
- Data (6 bytes)**: Data payload of the frame.
- Data: 61e263646566 [Length: 6]**: Hex dump of the data payload.
- No. Time Source Destination Protocol Length Info**: Column headers for the packet list.
- Packets List**: A list of 11 captured packets. The 4th packet is highlighted in red. The 11th packet is also highlighted in red.
- Hex View**: A hex dump of the selected packet (Frame 4). The data bytes are shown in pairs: 6000 00 04 00 01 00 00 06 08 00 27 11 b4 29 00 00 08 00 0019 45 00 00 3a 38 bd 40 00 40 06 e7 f8 aa 00 07 05 0020 84 00 07 04 c8 d8 2e e9 c5 a0 3d 3b b7 67 7f a5 0030 80 18 00 e5 22 35 00 00 01 01 08 0a 00 29 10 85 0040 00 00 62 1f 01 02 03 04 05 06
- Dec View**: A decimal dump of the selected packet.
- ASCII View**: An ASCII dump of the selected packet.
- Packets: 11 · Displayed: 11 (100.0%)**: Status bar at the bottom.
- Profile: Default**: Profile settings.

1. Suppose you run TCPClient before you run TCPServer. What happens? Why?

Ans :

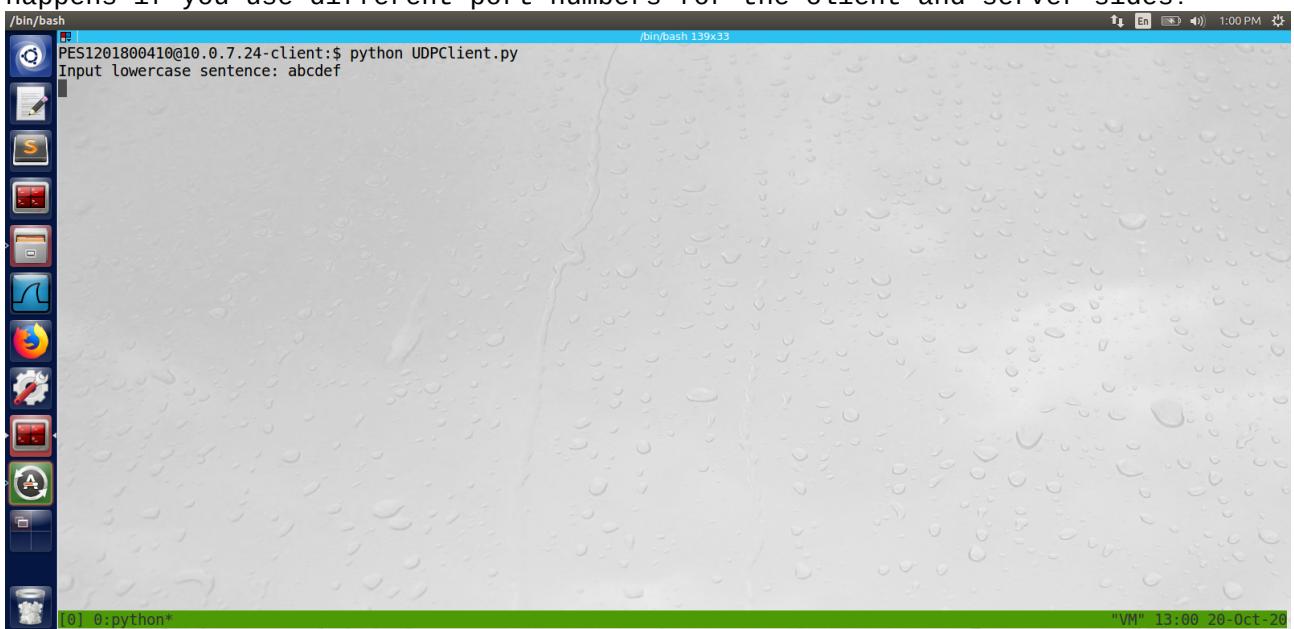


PES1201800410@10.0.7.24-client:\$ python TCPClient.py
Traceback (most recent call last):
File "TCPClient.py", line 5, in <module>
clientSocket.connect((serverName, serverPort))
File "/usr/lib/python2.7/socket.py", line 228, in meth
return getattr(self.sock,name)(*args)
socket.error: [Errno 111] Connection refused
PES1201800410@10.0.7.24-client:\$

[0] 0:bash* "VM" 12:56 20-Oct-20

we get a connection refused error because there is no server running to send/acknoledge the request sent by the tcp client

2. Suppose you run UDPClient before you run UDPServer. What happens? Why?
3.What happens if you use different port numbers for the client and server sides?



/bin/bash
PES1201800410@10.0.7.24-client:\$ python UDPClient.py
Input lowercase sentence: abcdef

[0] 0:python* "VM" 13:00 20-Oct-20

We wont receive a reply back because there is no server to send the response as it is a udp connection it doesn't really check if the server is running or in close state.

3.What happens if you use different port numbers for the client and server sides?

Same as the above answers tcp client will raise a connection refused error and udp client waits forever for the response to come form the server(which is running in different port so there won't be any reply)

Task 4 Mail Client

```
from socket import *
import ssl
import base64

# Message to send
msg = '\r\nI love computer networks!'
endmsg = '\r\n.\r\n'

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = 'smtp.gmail.com'

# Create socket called clientSocket and establish a TCP connection with
# mailserver
clientSocket = socket(AF_INET, SOCK_STREAM)

# Port number may change according to the mail server
clientSocket.connect((mailserver, 587)) #465 for ssl 587 for tls
recv = clientSocket.recv(1024)
print recv
if recv[:3] != '220':
    print '220 reply not received from server.'

# Send HELO command and print server response.
helocommand = 'HELO gmail.com\r\n'
clientSocket.send(helocommand)
recv1 = clientSocket.recv(1024)
print recv1
if recv1[:3] != '250':
    print '250 reply not received from server.'


clientSocket.send('starttls\r\n')
recv1 = clientSocket.recv(1024)
print recv1

secureConnect = ssl.wrap_socket(clientSocket, ssl_version=ssl.PROTOCOL_SSLv23)

secureConnect.send(helocommand)
recv1 = secureConnect.recv(1024)
print recv1

secureConnect.send('AUTH LOGIN\r\n')
recv1 = secureConnect.recv(1024)

print recv1

Username = 'prashanthathunt@gmail.com'
Password = "xxxxxxxxxxxxxxxxxxxxxx"

#print base64.b64encode(Username) + " username"
#print base64.b64encode(Password) + " Password"
secureConnect.send(base64.b64encode(Username) + '\r\n')
secureConnect.send(base64.b64encode(Password) + '\r\n')

recv_auth = secureConnect.recv(1024)
print recv_auth

print secureConnect.recv(1024)
```

```
# Send MAIL FROM command and print server response.
mailfrom = 'MAIL FROM: <prashanthathunt@gmail.com>\r\n'
secureConnect.send(mailfrom)
recv2 = secureConnect.recv(1024)
print recv2
if recv2[:3] != '250':
    print '250 reply not received from server.'

# Send RCPT TO command and print server response.
rcptto = 'RCPT TO: <bob@yahoo.com>\r\n'
secureConnect.send(rcptto)
recv3 = secureConnect.recv(1024)
print recv3
if recv3[:3] != '250':
    print '250 reply not received from server.'

# Send DATA command and print server response.
data = 'DATA\r\n'
secureConnect.send(data)
recv4 = secureConnect.recv(1024)
print recv4
if recv4[:3] != '354':
    print '354 reply not received from server.'

# Send message data.
secureConnect.send('SUBJECT: Greeting To you!\r\n')
secureConnect.send('test again')
secureConnect.send(msg)

# Message ends with a single period.
secureConnect.send(endmsg)
recv5 = secureConnect.recv(1024)
print recv5
if recv5[:3] != '250':
    print '250 reply not received from server.'

# Send QUIT command and get server response.
quitcommand = 'QUIT\r\n'
secureConnect.send(quitcommand)
recv6 = secureConnect.recv(1024)
print recv6
if recv6[:3] != '221':
    print '221 reply not received from server.'

secureConnect.close()
clientSocket.close()
```

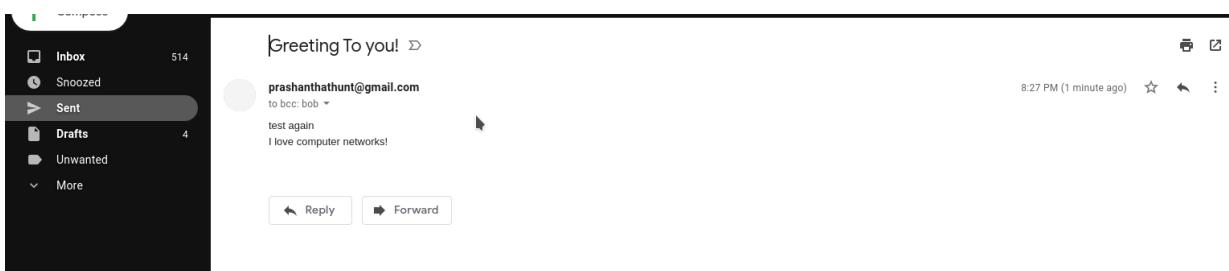
```

/bin/bash
PES1201800410@10.0.7.24-server$ python SMTPClient.py
220 smtp.gmail.com ESMTP y137sm3117385pfc.77 - gsmtp
250 smtp.gmail.com at your service
220 2.0.0 Ready to start TLS
250 smtp.gmail.com at your service
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 2.7.0 Accepted
250 2.1.0 OK y137sm3117385pfc.77 - gsmtp
250 2.1.5 OK y137sm3117385pfc.77 - gsmtp
354 Go ahead y137sm3117385pfc.77 - gsmtp
250 2.0.0 OK 1603225917 y137sm3117385pfc.77 - gsmtp
221 2.0.0 closing connection y137sm3117385pfc.77 - gsmtp
PES1201800410@10.0.7.24-server$ 

```

[0] 0:bash- 1:python* "VM" 16:32 20-Oct-20

cross check in gmail using web browser



wireshark output

Capturing From any

No.	Time	Source	Destination	Protocol	Length	Info
26	2020-10-20 16:24:37.3927640...	172.253.118.108	10.0.7.4	SMTP	108	S: 220 smtp.gmail.com ESMTP t13sm3014305pfc.1 - gsmtp
27	2020-10-20 16:24:37.3929965...	10.0.7.4	172.253.118.108	TCP	56	55760 - 587 [ACK] Seq=2843204692 Ack=7289 Win=29200 Len=0
28	2020-10-20 16:24:37.3940572...	10.0.7.4	172.253.118.108	SMTP	72	C: HELO gmail.com
29	2020-10-20 16:24:37.5749858...	172.253.118.108	10.0.7.4	TCP	62	587 - 55760 [ACK] Seq=7289 Ack=2843204708 Win=32752 Len=0
30	2020-10-20 16:24:37.6996833...	172.253.118.108	10.0.7.4	SMTP	92	S: 250 smtp.gmail.com at your service
31	2020-10-20 16:24:37.6998309...	10.0.7.4	172.253.118.108	SMTP	66	C: starttls
32	2020-10-20 16:24:37.8254715...	172.253.118.108	10.0.7.4	TCP	62	587 - 55760 [ACK] Seq=7325 Ack=2843204718 Win=32742 Len=0
33	2020-10-20 16:24:38.0072563...	172.253.118.108	10.0.7.4	SMTP	86	S: 220 2.0.0 Ready to start TLS
34	2020-10-20 16:24:38.0090577...	10.0.7.4	172.253.118.108	TLSV1.2	573	Client Hello
35	2020-10-20 16:24:38.0761261...	172.253.118.108	10.0.7.4	TCP	62	587 - 55760 [ACK] Seq=7355 Ack=2843205235 Win=32225 Len=0
36	2020-10-20 16:24:38.0811855...	172.253.118.108	10.0.7.4	TLSV1.2	1414	Server Hello
37	2020-10-20 16:24:38.1236088...	10.0.7.4	172.253.118.108	TCP	56	55760 - 587 [ACK] Seq=2843205235 Ack=8713 Win=31234 Len=0
38	2020-10-20 16:24:38.1239097...	172.253.118.108	10.0.7.4	TLSV1.2	1271	Certificate, Server Key Exchange, Server Hello Done
39	2020-10-20 16:24:38.1239499...	10.0.7.4	172.253.118.108	TCP	56	55760 - 587 [ACK] Seq=2843205235 Ack=9928 Win=33950 Len=0
40	2020-10-20 16:24:38.1279266...	10.0.7.4	172.253.118.108	TLSV1.2	182	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
41	2020-10-20 16:24:38.1936400...	172.253.118.108	10.0.7.4	TLSV1.2	343	New Session Ticket, Change Cipher Spec, Hello Request, Hello Response

Frame 30: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface
 ► Linux cooked capture
 ► Internet Protocol Version 4, Src: 172.253.118.108, Dst: 10.0.7.4
 ► Transmission Control Protocol, Src Port: 587, Dst Port: 55760, Seq: 7285
 ▾ Simple Mail Transfer Protocol
 ▾ Response: 250 smtp.gmail.com at your service\r\n

Response code: Requested mail action okay, completed (250)
 Response parameter: smtp.gmail.com at your service

0000 00 00 00 01 00 06 52 54 00 12 35 00 00 00 00 00RT ..5....
 0010 45 00 00 04 00 99 00 00 ff 06 86 ae ac fd 0c 6c E..L....vl
 0020 00 00 07 64 02 4b d0 d0 00 00 1c 79 a9 77 dd 64K.y.w.d
 0030 50 18 7f f8 9f 7e 00 00 32 35 39 20 73 6d 74 76 P.....~. 250 smtp
 0040 28 07 60 01 69 6c 20 63 67 00 20 61 74 20 79 0f .gmail.c om at yo
 0050 75 72 20 73 65 72 69 63 65 00 0a ur serv ice..

Packets: 128 · Displayed: 128 (100.0%) Profile: Default