NETWORK SECURITY

Assignment 1

Soheil Shirvani 3720505

Introduction

In this assignment, we are first going to run 2 VMs, namely Linux Kali Attacker and Windows Victim VMs. Then we run a server-side socket program written by instructor in dotnet on Windows Victim VM and we are going to program a client-side socket on Kali Linux Attacker VM. After making the connection between client and server-side socket, we are going to send our name and student number via the socket to from client side to the server side.

For detail analysis, we have run “netstat -an” to see if the sockets are listening and to find the state of each socket in both client and server side.

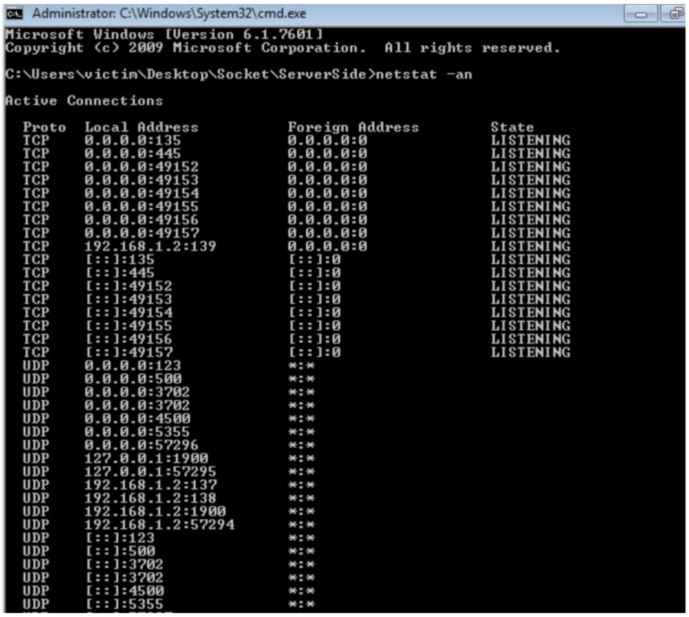
For further analysis we run Wireshark as a network analysis tool to find out the packet we sent earlier and to see the plain text payload as it has no security defined on top of our socket.

All the screenshots, Codes, and analysis are here in this report.

Analysis

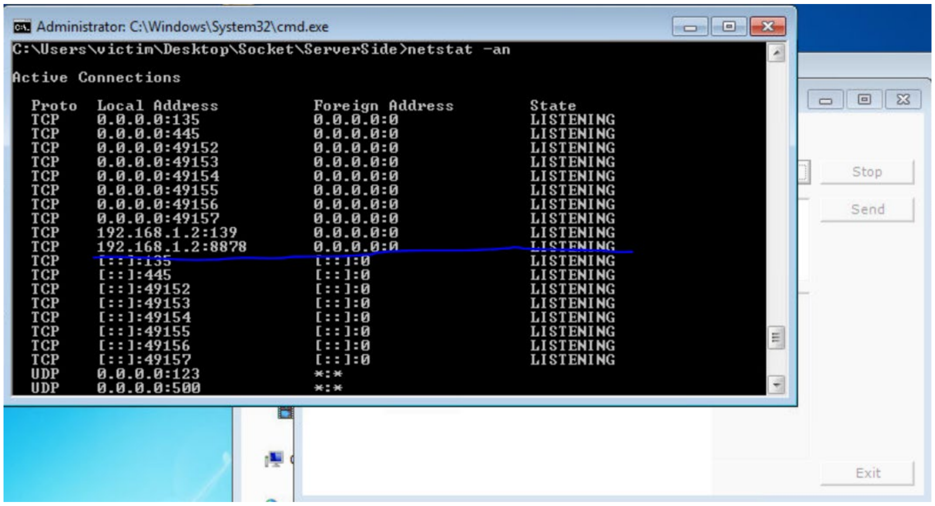
Firstly, I am going to run server side socket on the Windows Victim VM and to make sure that it is running I am going to run 'netstat -an' before and after running socket on CMD.

Before we run the socket we can see:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\victim\Desktop\Socket\ServerSide>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49156          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49157          0.0.0.0:0              LISTENING
  TCP    192.168.1.2:139        0.0.0.0:0              LISTENING
  TCP    [::]:135               [::]:0                 LISTENING
  TCP    [::]:445               [::]:0                 LISTENING
  TCP    [::]:49152             [::]:0                 LISTENING
  TCP    [::]:49153             [::]:0                 LISTENING
  TCP    [::]:49154             [::]:0                 LISTENING
  TCP    [::]:49155             [::]:0                 LISTENING
  TCP    [::]:49156             [::]:0                 LISTENING
  TCP    [::]:49157             [::]:0                 LISTENING
  UDP    0.0.0.0:123            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:3702           *:*
  UDP    0.0.0.0:3702           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    0.0.0.0:5355           *:*
  UDP    0.0.0.0:57296          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    127.0.0.1:57295        *:*
  UDP    192.168.1.2:137        *:*
  UDP    192.168.1.2:138        *:*
  UDP    192.168.1.2:1900       *:*
  UDP    192.168.1.2:57294      *:*
  UDP    [::]:123               *:*
  UDP    [::]:500               *:*
  UDP    [::]:3702              *:*
  UDP    [::]:3702              *:*
  UDP    [::]:4500              *:*
  UDP    [::]:5355              *:*
```

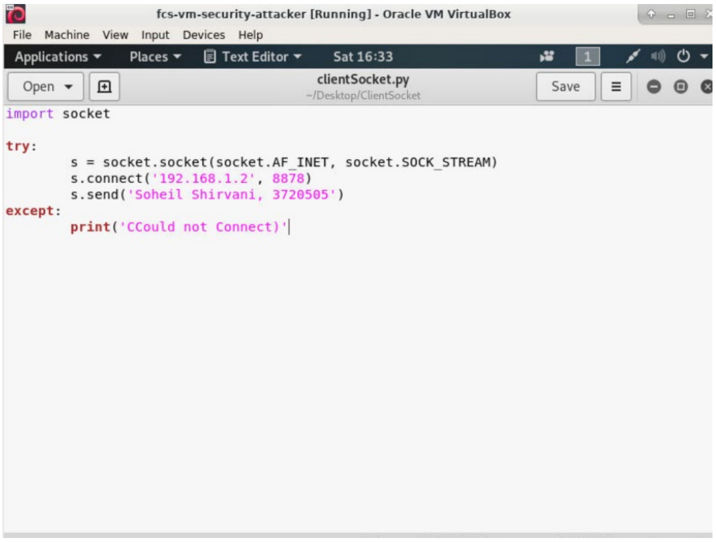
There is no defined socket in here that is listening to a specific port. Then I run server side socket program with the ip of the computer which is '192.168.1.2' and port '8878'. Then I run 'netstat -an' again and the result is:



```
C:\Users\victim\Desktop\Socket\ServerSide>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49156          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49157          0.0.0.0:0              LISTENING
  TCP    192.168.1.2:139        0.0.0.0:0              LISTENING
  TCP    192.168.1.2:8878       0.0.0.0:0              LISTENING
  TCP    [::]:135               [::]:0                 LISTENING
  TCP    [::]:445               [::]:0                 LISTENING
  TCP    [::]:49152             [::]:0                 LISTENING
  TCP    [::]:49153             [::]:0                 LISTENING
  TCP    [::]:49154             [::]:0                 LISTENING
  TCP    [::]:49155             [::]:0                 LISTENING
  TCP    [::]:49156             [::]:0                 LISTENING
  TCP    [::]:49157             [::]:0                 LISTENING
  UDP    0.0.0.0:123            *:*
  UDP    0.0.0.0:500            *:*
```

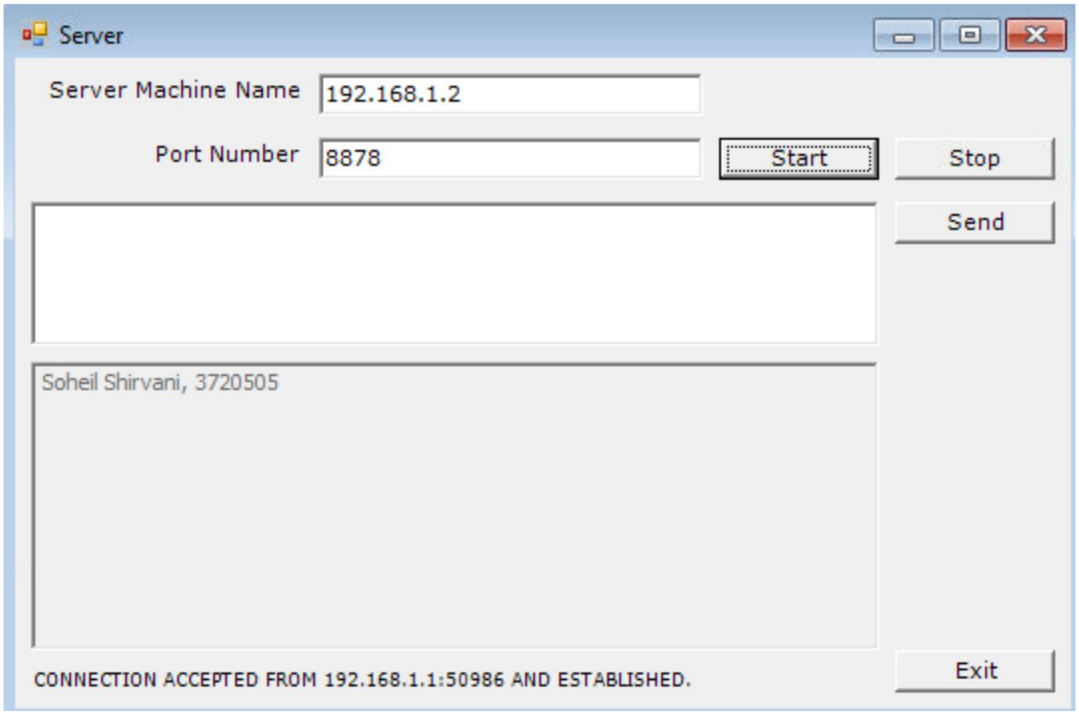
We can see the socket is listening on port 8878. This was for the server side socket.

Now I am going to define client socket program on Kali Linux Attacker VM. To do so, I have used python and wrote a very simple client socket program to send my name and student number to the windows machine. The program looks below:


fcs-vm-security-attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Text Editor Sat 16:33
Open
clientSocket.py
~/Desktop/ClientSocket
Save

```
import socket

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect('192.168.1.2', 8878)
        s.send('Soheil Shirvani, 3720505')
except:
        print('CCould not Connect)'
```

The program is simple. It creats a socket on the first line and then connect the socket to the server which was on ip ‘192.168.1.2’ on port ‘8878’ which we defined earlier on server side. Then the socket will send ‘Soheil Shirvani 3720505’ which includes my name and student number to the server socket. After running the client program we can see in the server:



Server Machine Name: 192.168.1.2

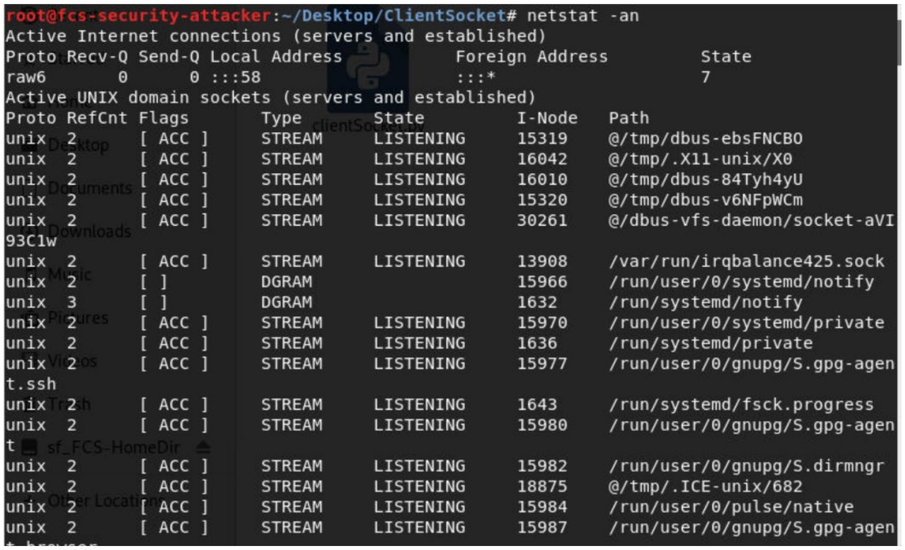
Port Number: 8878

```
Soheil Shirvani, 3720505
```

CONNECTION ACCEPTED FROM 192.168.1.1:50986 AND ESTABLISHED.

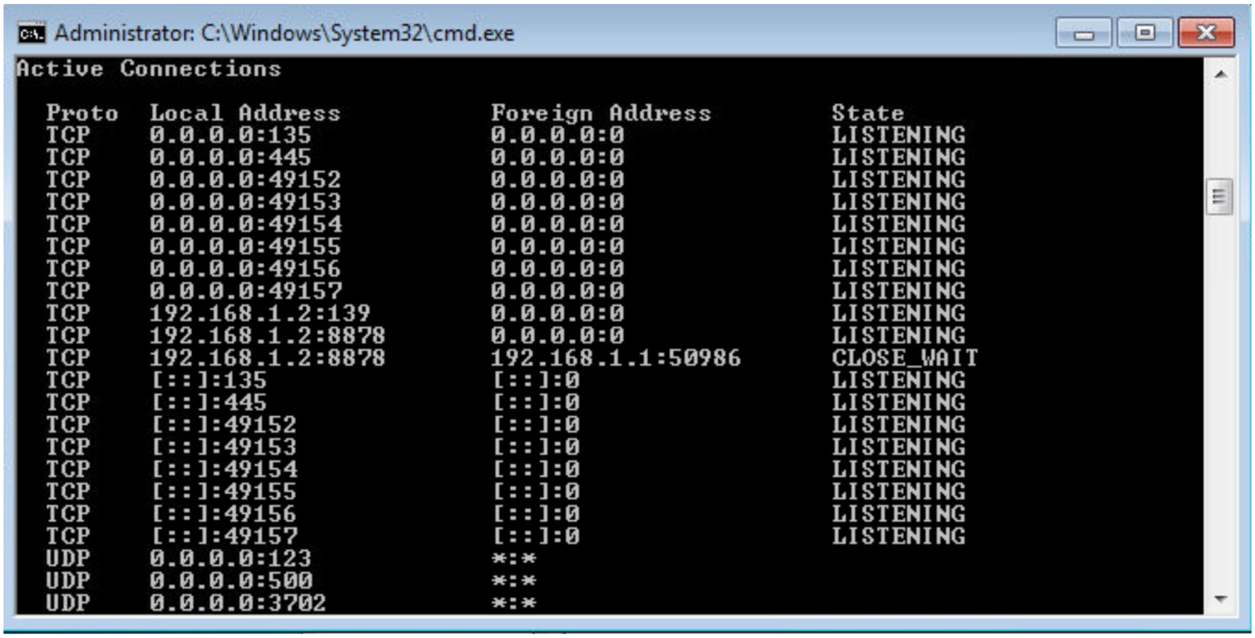
It can be seen in the figure that the message had been received and connection was successful from ip '192.168.1.1' which is the Kali Linux VM.

We ran 'netstat -an' to make sure everything works as expected in both client and the server. For the Client part before sending the message we can see:



```
root@fcs-security-attacker:~/Desktop/ClientSocket# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
raw6       0      0 :::58                   :::*                    7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     15319    @/tmp/dbus-ebsFNCBO
unix  2      [ ACC ]     STREAM     LISTENING     16042    @/tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     16010    @/tmp/dbus-84Tyh4yU
unix  2      [ ACC ]     STREAM     LISTENING     15320    @/tmp/dbus-v6NFpWCm
unix  2      [ ACC ]     STREAM     LISTENING     30261    @/dbus-vfs-daemon/socket-aVI
93C1w
unix  2      [ ACC ]     STREAM     LISTENING     13908    /var/run/irqbalance425.sock
unix  2      [ ]         DGRAM                    15966    /run/user/0/systemd/notify
unix  3      [ ]         DGRAM                    1632     /run/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     15970    /run/user/0/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     1636     /run/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     15977    /run/user/0/gnupg/S.gpg-agen
t.ssh
unix  2      [ ACC ]     STREAM     LISTENING     1643     /run/systemd/fsck.progress
unix  2      [ ACC ]     STREAM     LISTENING     15980    /run/user/0/gnupg/S.gpg-agen
t
unix  2      [ ACC ]     STREAM     LISTENING     15982    /run/user/0/gnupg/S.dirmngr
unix  2      [ ACC ]     STREAM     LISTENING     18875    @/tmp/.ICE-unix/682
unix  2      [ ACC ]     STREAM     LISTENING     15984    /run/user/0/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     15987    /run/user/0/gnupg/S.gpg-agen
```

There is no define socket. And After running the socket program and running netstat on server side we can see:



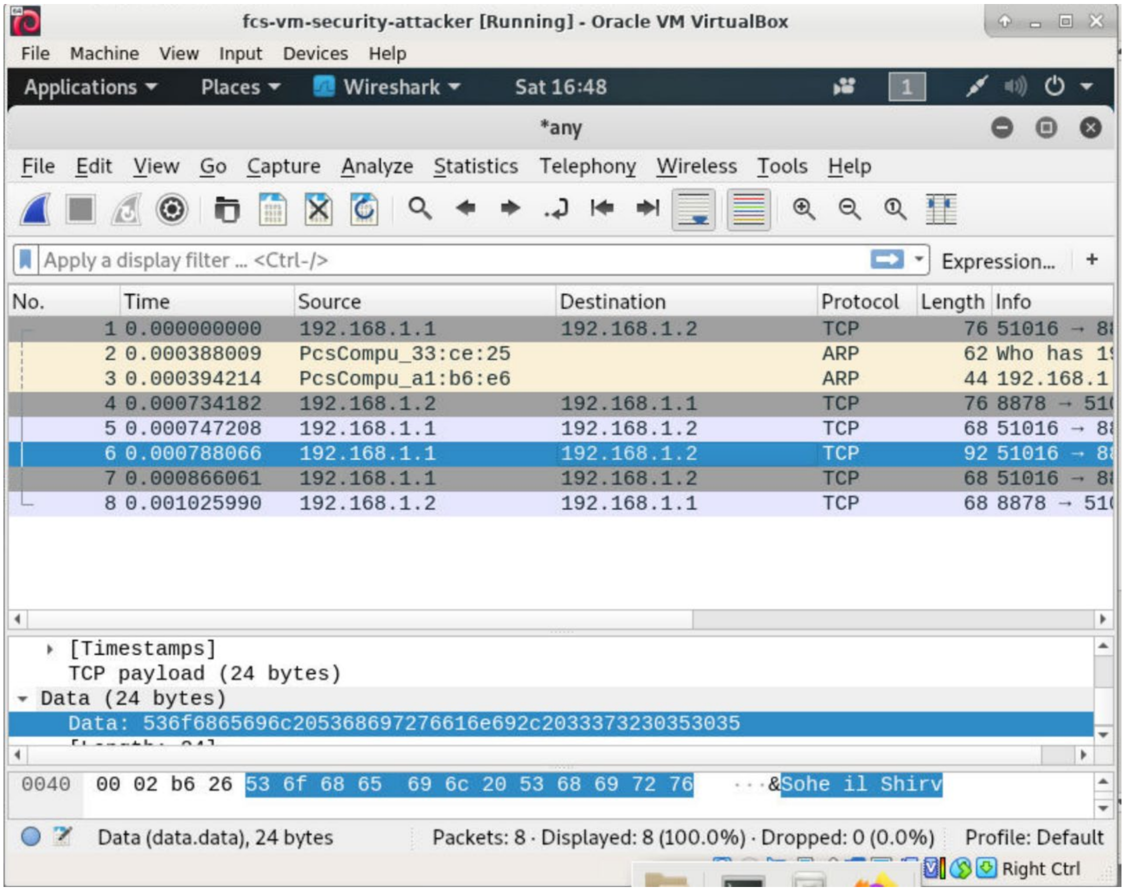
```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49156          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49157          0.0.0.0:0              LISTENING
  TCP    192.168.1.2:139        0.0.0.0:0              LISTENING
  TCP    192.168.1.2:8878       0.0.0.0:0              LISTENING
  TCP    192.168.1.2:8878       192.168.1.1:50986      CLOSE_WAIT
  TCP    [::]:135               [::]:0                 LISTENING
  TCP    [::]:445               [::]:0                 LISTENING
  TCP    [::]:49152             [::]:0                 LISTENING
  TCP    [::]:49153             [::]:0                 LISTENING
  TCP    [::]:49154             [::]:0                 LISTENING
  TCP    [::]:49155             [::]:0                 LISTENING
  TCP    [::]:49156             [::]:0                 LISTENING
  TCP    [::]:49157             [::]:0                 LISTENING
  UDP    0.0.0.0:123            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:3702           *:*
```

Here it can be seen that there was a connection between '192.168.1.2' which is server and '192.168.1.1' which is the client. A connection is close_wait because it already sends its message and then close the connection. There is also a similar IP on listening which is the server still looking for a new socket and messages. Client can run the socket again to send more messages.

For further analysis we run wireshark to see the message.

After running Wireshark we start to look for the packets on ‘any’ interface. And the I run client socket again to capture the packet.

This is the packets after running client socket program:


fcs-vm-security-attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark Sat 16:48
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Right Ctrl

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.1.2	TCP	76	51016 → 8
2	0.000388009	PcsCompu_33:ce:25		ARP	62	Who has 1
3	0.000394214	PcsCompu_a1:b6:e6		ARP	44	192.168.1
4	0.000734182	192.168.1.2	192.168.1.1	TCP	76	8878 → 51
5	0.000747208	192.168.1.1	192.168.1.2	TCP	68	51016 → 8
6	0.000788066	192.168.1.1	192.168.1.2	TCP	92	51016 → 8
7	0.000866061	192.168.1.1	192.168.1.2	TCP	68	51016 → 8
8	0.001025990	192.168.1.2	192.168.1.1	TCP	68	8878 → 51

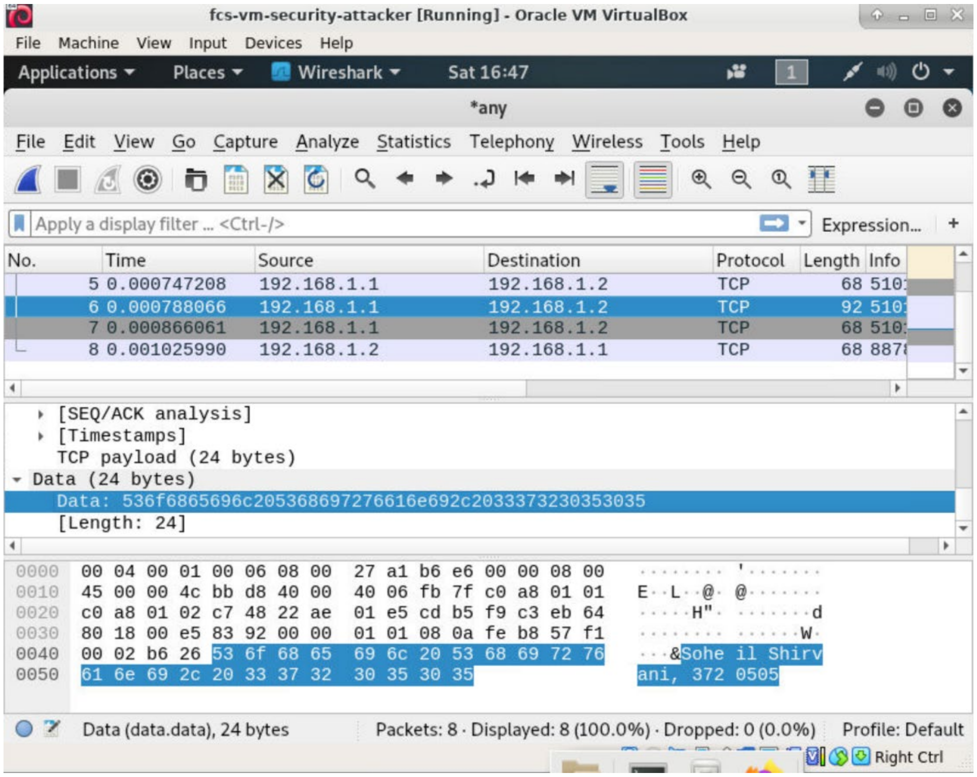
```
  ▸ [Timestamps]
    TCP payload (24 bytes)
▾ Data (24 bytes)
    Data: 536f6865696c205368697276616e692c2033373230353035
```

```
0040   00 02 b6 26 53 6f 68 65  69 6c 20 53 68 69 72 76     ···&Sohe il Shirv
```

Data (data.data), 24 bytes   Packets: 8 · Displayed: 8 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

It can be seen that there are a couple of packets sent to the client and received from client. Firstly the first packet is that client sends server a request regarding a connection to the socket. The second packet is that server ACK the packet and sends it to the client. Then client sends a packet regarding that It want to sends data to the server and the immediately after that it sends the packet with payload data to the server. After completion of the packet Client again sends a packet informing the server that the payload is finished and data has been sent. In this period the server stat have changed from ‘LISTENING’ to ‘CONNECTION’. And then server replys with an ACK that it received the packet from the CLIENT.

Also in the fourth packet we can see the data payload in plain text as below:


fcs-vm-security-attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark Sat 16:47
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time          Source          Destination     Protocol  Length  Info
    5 0.000747208   192.168.1.1     192.168.1.2     TCP           68  510
    6 0.000788066   192.168.1.1     192.168.1.2     TCP           92  510
    7 0.000866061   192.168.1.1     192.168.1.2     TCP           68  510
    8 0.001025990   192.168.1.2     192.168.1.1     TCP           68  887
```

```
    [SEQ/ACK analysis]
    [Timestamps]
    TCP payload (24 bytes)
  Data (24 bytes)
    Data: 536f6865696c2053686972766616e692c2033373230353035
    [Length: 24]
```

```
0000  00 04 00 01 00 06 08 00  27 a1 b6 e6 00 00 08 00   ........ '.......
0010  45 00 00 4c bb d8 40 00  40 06 fb 7f c0 a8 01 01   E··L··@· @·······
0020  c0 a8 01 02 c7 48 22 ae  01 e5 cd b5 f9 c3 eb 64   ·····H"· ·······d
0030  80 18 00 e5 83 92 00 00  01 01 08 0a fe b8 57 f1   ········ ······W·
0040  00 02 b6 26 53 6f 68 65  69 6c 20 53 68 69 72 76   ···&Sohe il Shirv
0050  61 6e 69 2c 20 33 37 32  30 35 30 35               ani, 372 0505
```

Data (data.data), 24 bytes   Packets: 8 · Displayed: 8 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

Plain text of the data can be seen on the right bottom corner. We can see the plain text of the data because no security feature had been used to encrypt the data.

Socket States

Here I am going to talk about different socket states in server side and client side. First I am going to talk about the server side:

1. Create Socket: Server first create a socket for connection within its IP.
2. Bind the Socket: Then server tries to bind the PORT address that it is going to listen on to the socket so others can connect to it.
3. Listen for CLIENT: The first state of the server socket after creating is LISTENING. In this state server socket is waiting for a client to connect to the socket the server is listening on. This state is waiting state for server.
4. Accept Connection: After a client request to connect to the socket, the server sends an accept packet to the client and change its state from 'LISTENING' to the second state 'CONNECTED/ESTABLISHED' which shows there is a connection established between the server and the client.
5. Receive Request: Then the client which is already connected to the server socket can send request to the server. These requests can be a request to send data, request to send command or any other request. Server receives the Request and send a message that it agrees or not (ACK/DNY)
6. Send Response: After accepting clients request client will sends its request and data to the server. Server response with an ACK that it received the request and data and confirming it.
7. CLOSING: Client can close the socket connection after its done with the server or server can close the connection after a timeout. This way the server state changes from 'CONNECTION' to 'CLOSE_WAIT' which means the client clint close the socket but sever is still 'Listening' so server close that socket and OPENs another socket with the state of 'LISTENING' which is the '4' stage that was described here. Or server can Close the socket entirely too which the state changes to 'CLOSED' and server is no longer listening for a client

The Client Side states:

1. Create Socket: Firstly, the Client creates a socket which tries to connect to a remote server.
2. Connect to Server: Client tries to connect to an IP address and a PORT of the server, that server already tries to LISTEN for the client. Server will send ACK if it can receive the client connection. The state of Client socket changes to 'CONNECTION/ESTABLISHED'
3. Send Request: Client sends its request to the server and Server sends ACK to Client. Then Client can send the request and the data to the server.
4. Receive Response: Server after getting the message from client that the message is finished and request had been sent, sends an ACK to the client that the message had been received. Client they can stay on the socket for further messages which is stage 3 or close the connection. In the closing form, client socket state changes to 'CLOSED' and socket will be closed until client create another socket which is stage 1.