Cryptography

.

Assignment3 Report

.

Vulnerabilities in standard cryptographic protocols

Anant Jain - (2019MCS2557)

August 14, 2020

Introduction

In this assignmet , I have implemented **Heartbleed** vulnerability of TLS/SSL protocol.

1 TLS/SSL protocol :-

After the coming of the web , SSL(Secure Socket Layers) protocol is introduced in 1996 to provide secure communication.TLS (Transport Layer Security) is the upgraded version of SSL protocol, introduced by Internet Engineering Task Force (IETF) in 1999.
TLS/SSL protocol are used to provide security,privacy and integrity on computer network.They are used in many applications like instant messaging, email , web browsing etc. They are one of the most widely used protocol over computer network in the world.

1.1 Working of the TLS/SSL protocol:-

As, for communication over internet , Encryption is required to have security and privacy.As, Asymmetric Key Encryption (Public and Private key encryption) is more secure than any other method.And, TLS/SSL protocol uses asymmetric encryption.

At the beginning of the connection between the client and server, the handshake process happen.In this ,by use of asymmetric cryptography , both server and client agree to make a shared session key which (i.e. a type of symmetric communication) is used for communication in rest of the entire session.By this, communication between them is now secure.

1.2 TLS/SSL handshake process :-

The TLS/SSL handshake process has following steps :-

- The client will send required information to server for starting the HTTPS connection. The server send its reply to client. After that, Client sends certificate request to the server.
- The sender send its certificate to the client and client sends its certificate to the server. And they establish connection between them by making a session key.

- After the connection establishment, server and client communicate with each other.

2 Vulnerability taken :- Heartbleed

Though, TLS/SSL protocol have lot of vulnerability like POODLE,BEAST,CRIME,BREECH etc,but Heartbleed is very critical vulnerability. Attacker can exploit it many times without leaving any trace.

It is a security bug introduced in the TLS protocol in 2012 and is disclosed in 2014.Due to this ,lot of theft of data take place.Thousands of websites and internet devices are vulnerable to this bug.

2.1 Explaination of Heartbleed bug in TLS:-

The heartbeat extension was introduced in TLS protocol to check whether the connections are alive or not.But, this extension has a serious problem which causes this bug.As, this bug is a result of heartbeat extension, it get the name of heartbleed bug.

Heartbleed bug is a result of implementation fault in OpenSSL cryptography library,widely used in the TLS protocol. It occur due to lack of valid check when heartbeat request is sent. The client send heartbeat message and heartbeat message length. Here, length of heartbeat message is not checked against the inputted message length. The inputted message is stored in server memory by server. And,The server response to client with the message of inputted heartbeat message length (it contain inputted heartbeat message and confidential information of server).By this, client get the confidential data of server memory.

2.2 Diagram showing Heartbleed bug :-

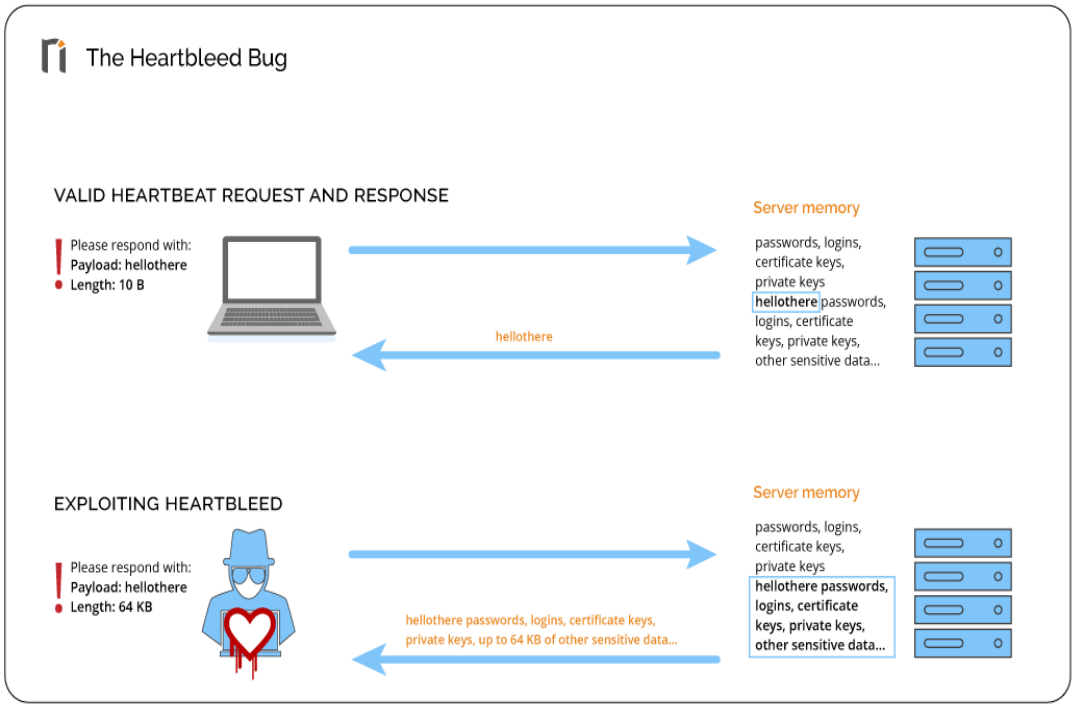

Figure 1: Diagram showing Heartbleed bug

2.3 Companies affected :-

Companies affected by this vulnerabilities are Cisco, Yahoo ,Stack Overflow , DuckDuckGo , Reddit , Tumblr ,Amazon web services etc.

2.4 Prevention :-

- Upgrade version of OpenSSL Library
- Always Perform server side input validation.
- Never trust input from external system without authenticating them.
- Before using any protocol or application, check its security review to know about its any vulnerabilities.

3 How to exploit this bug :-

- When Attacker connect with the server.He send a heartbeat request. The heartbeat request consist of heartbeat message and heartbeat message length. Here, Length of heartbeat message sent by attacker is less than inputted heartbeat message length. Due to this, when the server get this message, the server donot validate the message length of input heartbeat message and input heartbeat message length.
- Now, Server store the input heartbeat message in the server memory which has lot of confidential important information in plain text. Here, Server has to response the inputted message to the client of given inputted heartbeat message length.
- Since , the two lengths are not same. Server take content of inputted heartbeat message length (it also include the inputted heartbeat message) and sent it to attacker(client)
- Here, Client can send a message of very small bytes and input message length of 65,536 length (i.e.64kb). So, By this bug,Attacker can get lot of data from server without being traced. Also, The attacker can exploit this bug any number of times.

4 Explaination of code :-

4.1 Few point on the implementaion of code:-

- In this code, I have used librarirs like gmpy2,random, socket and threading.
- In this code, I have written code in 3 python files :- client_hb.py , server_hb.py and KeyGenForSerAndClt.py.
- Also, After handshaking between client and server, they can communicate with each other.Here, I have assumed that channe is secure for communication So. encryption and decryption is not used.Message from client is sent directly to server and viceversa.

- I have given additional functionality for server that it can view content of server memory on terminal by use of viewBufferContent function and can insert content in the server memory or buffer by use of insertContentInSerBuffer function on terminal.
- Here, Communication between server and client is such that no one can give input to terminal consecutively. After client send a request , then server have to response back to client and vice-versa.
- Communication start from client side initially.
- For Convenience, It is assumed that only client can close the connection.

4.2 Python files in this implementation :-

- **KeyGenForSerAndClt.py :-** It is used to produce public and private key of the server and client.
  - **generate_prime :-** Given count of bits, it is used to produce prime number of given count of bits.
  - **generate_key_content:-** Given value of phi, it is used to produce content of public and private key (i.e. e and d)
- **server_hb.py :-** In this file, all the functions are implemented that will be used by server application.
  - **sendHeartBeatResponse :-** Given server socket and heartbeat message by client, it is used to give heartbeat response to the client.
  - **insertContentInSerBuffer :-** Given data by the server side terminal , it it used to insert inputted data into server memory or buffer of server.(NOTE:- this fuction is built for testing purposes)
  - **viewBufferContent :-** This function is used by the server side terminal for viewing content of server side (NOTE:- this fuction is built for testing purposes)
  - **server_process :-** This fuction is called by main fuction of server file to implement server functionality.
- **client_hb.py** :- In this file, all the functions are implemented that will be used by client application.
  - **sendClientHelloToServer :-** This function is used to send hello message to server by client socket during handshaking.
  - **decodeAndSplit :-** In this, data sent by server socket is processed.
  - **getServerHello :-** It return true if server has send hello message to client during handshaking , otherwise false.
  - **sendCertiReqToServer :-** By this, client socket send server certificate request to server during handshaking.
  - **getServerCertificate :-** It return true and server certificate if, server send its certificate to client, otherwise false.

- **sendCertificateToServer :-** In this,Client socket send its certificate to server.
- **getFinishHandshakeRequestFromServer :-** In this, client socket receive request of finishing handshake from server.
- **sendFinishHandShakeRequestToServer :-** In this, client socket send request to finish hand shake request to server.
- **getServerReadyMessage :-** It return true if client receive server ready message from socket.
- **handshakeByServer :-** This function call all above function sequentially to make connection between server and client.
- **client_process :-** This fuction is called by main function of client file to implement client functionality.

4.3 Commands used by server and client :-

4.3.1 By Server :-

- **message** :- Server can send message to client.
- **insertbuffer** :- Server (user who is at server side) can add content in the server memory or buffer.
- **viewbuffer** :- Server (user who is at server side) can view content of the server memory or buffer.

4.3.2 By Client :-

- **message** :- Client can send message to server.
- **heartbeat** :- By this, client can send heartbeat request (with heartbeat messge and heartbeat message length) to client.
- **close** :- By this, Client side application will be closed.(NOTE :- Server side application will be closed with some error message)

4.4 Procedure to run the python code :-

- Open two terminal in the directory in which code resides.
- On one terminal, run server file (i.e. python server_hb.py)
- On other terminal, run client file (i.e. python client_hb.py)
- Now , give heartbeat command on the client terminal.

4.5 Heartbleed code :-

4.5.1 For Server Side :-

```
def sendHeartBeatResponse(c,cldata):
cldata = cldata.split(" ")
datalen = int(cldata[-1])
cldata = cldata[:-1]
cldata = " ".join(cldata)

bufferfile = open("server_buffer.txt","r")
bufile = bufferfile.readlines()
bufferfile.close()

bufstrg = ''
for line in bufile:
    bufstrg = bufstrg + line

clmsglen = len(cldata)

if clmsglen < datalen:
    bufflen = datalen - clmsglen
    newstrg = cldata + bufstrg

    respstrg = cldata + bufstrg[0:bufflen]

    bufferfile = open("server_buffer.txt","w")
    bufferfile.write(newstrg)
    bufferfile.close()

    c.send(respstrg.encode())
else:
    c.send(cldata.encode())
```

4.5.2 For Client Side :-

```
elif commandInpRec == "heartbeat":

    heartbeatmsg = input('Enter your heartbeat message :-')
    heartbeatlength = input('Enter heartbeat message length :- ')
    clhbmsg = "heartbeat"
    clhbmsg = clhbmsg + " "+"02"
    clhbmsg = clhbmsg + " "+heartbeatmsg+" "+heartbeatlength
    print(' inputted heartbeat message is sent to sever. \n')
    client_socket.send(clhbmsg.encode())
    while True:
```

```
srhbmsg = client_socket.recv(1024)
srhbmsg = srhbmsg.decode()
if srhbmsg:
    print(' response of heartbeat message by server :- ',srhbmsg)
    print('\n')
    break
```

4.6 Screenshots of Terminal while running the code :-

4.6.1 Running Server file ,then running client file :-

Server side terminal :-

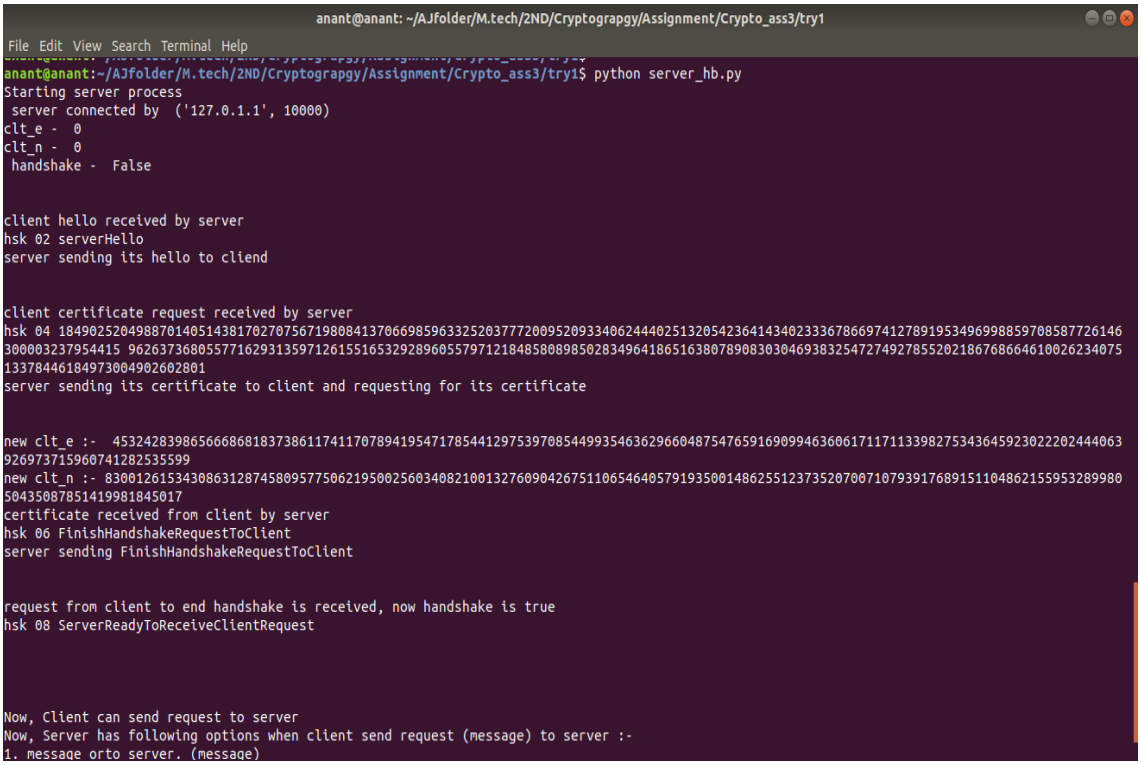

Figure 2: Server terminal after handshaking

Client side terminal :-

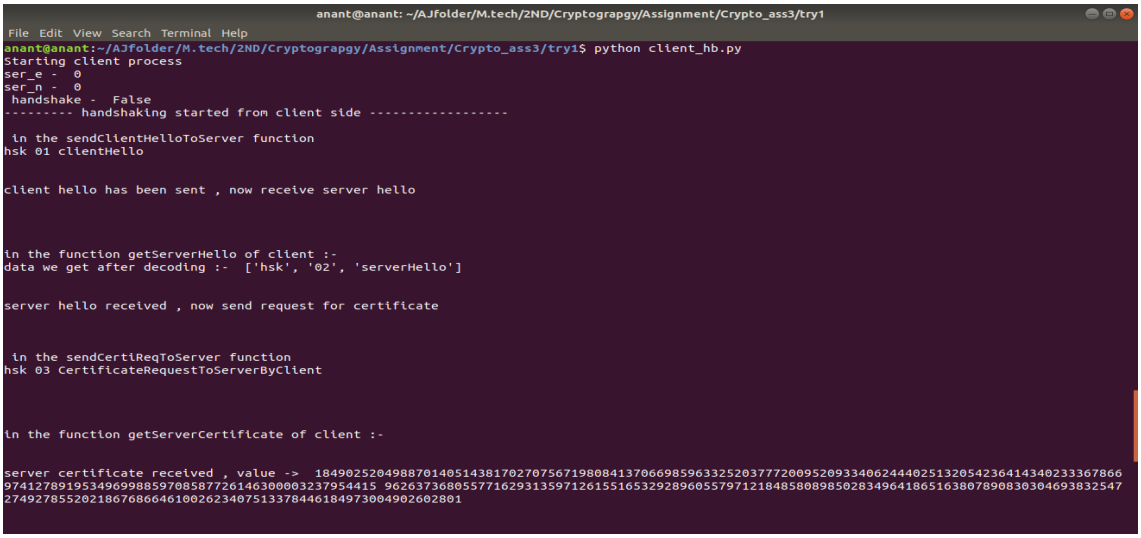

Figure 3: Client terminal after handshaking

4.6.2 Message command from client and the from server :-

Client side terminal :-

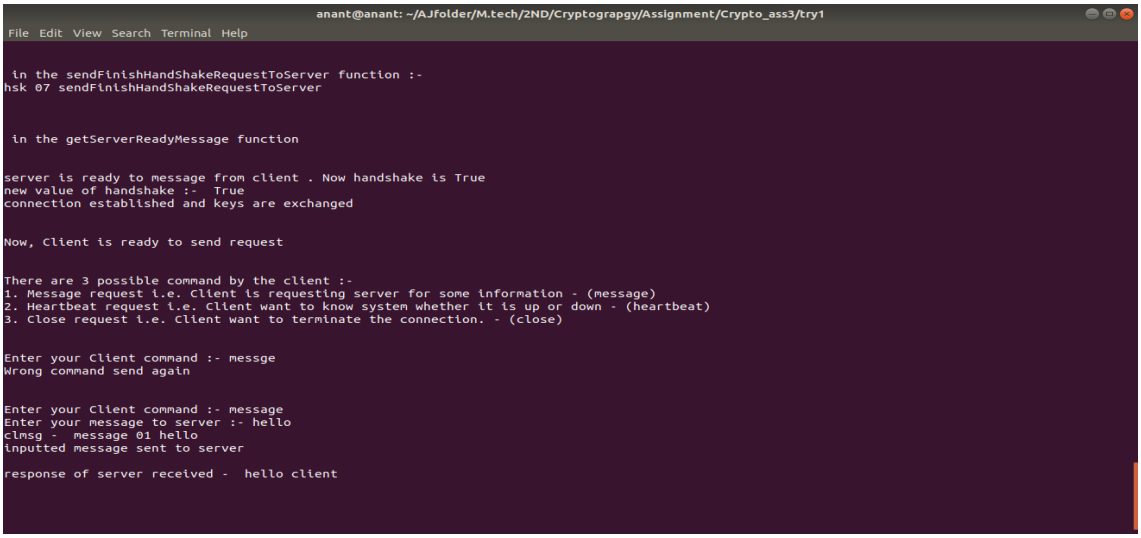

Figure 4: Client terminal after message command

Server side terminal :-

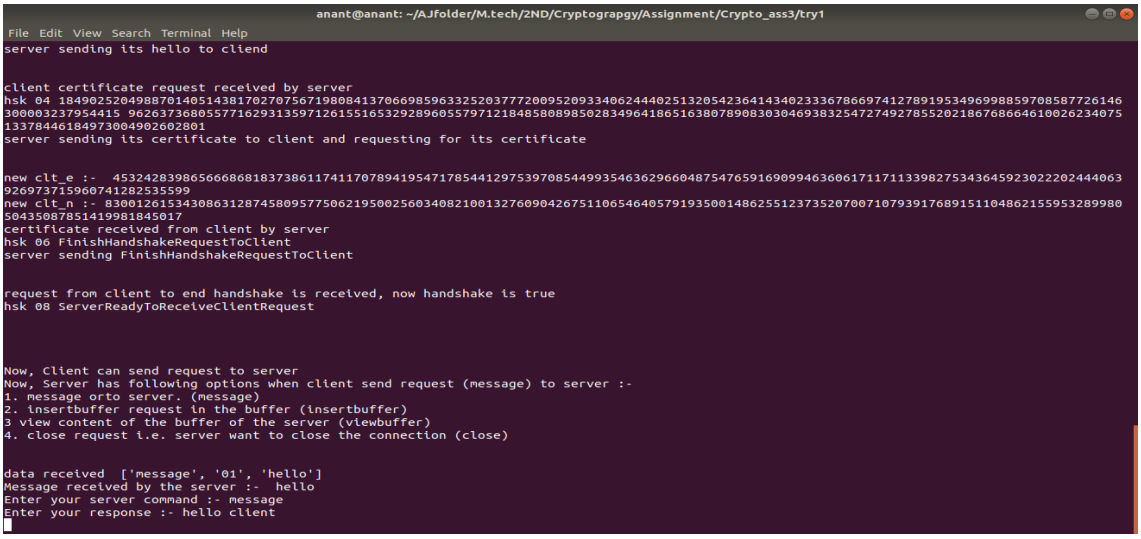

Figure 5: Client terminal after message command

4.6.3 Server Insert buffer command :-

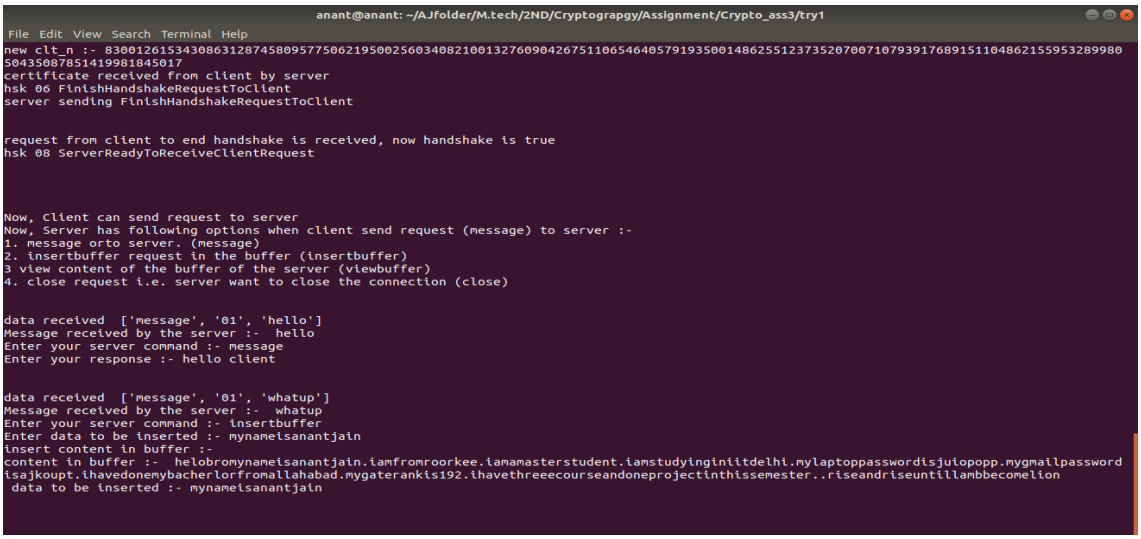

Figure 6: Server terminal after insertbuffer command

4.6.4 Server view buffer command :-

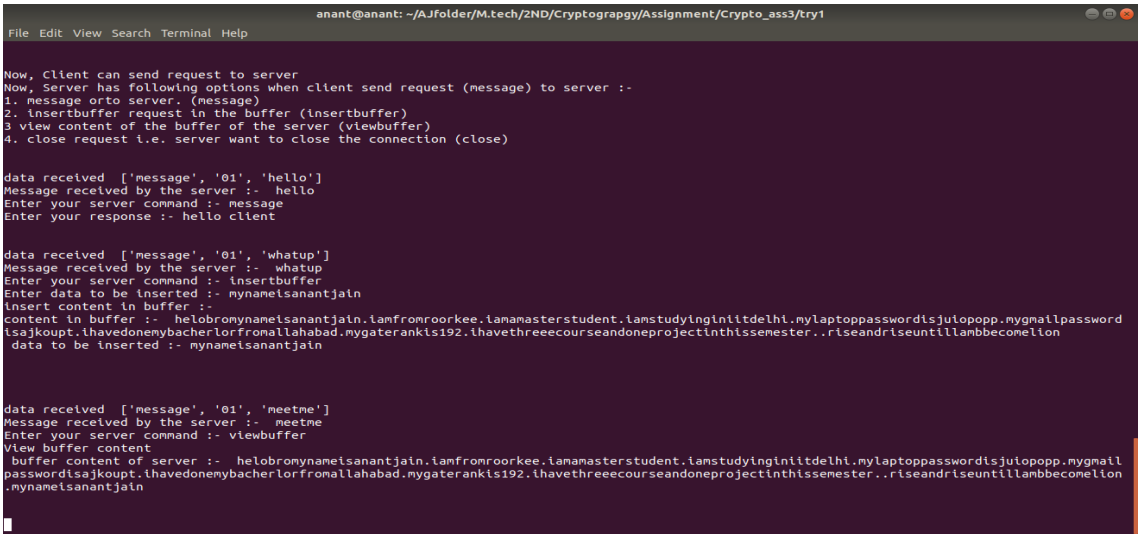

Figure 7: Server terminal after view command

4.6.5 Heartbleed vulnerability simulation :-

Client sending heartbeat request and getting its response:-

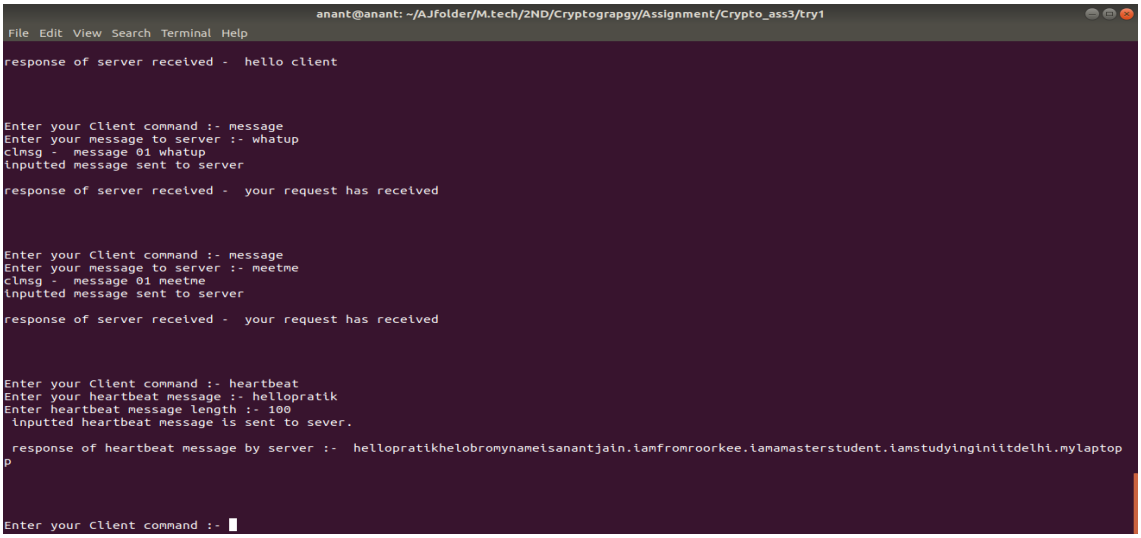

Figure 8: Client terminal after heartbeat request

Server responding to heartbeat request of Client :-

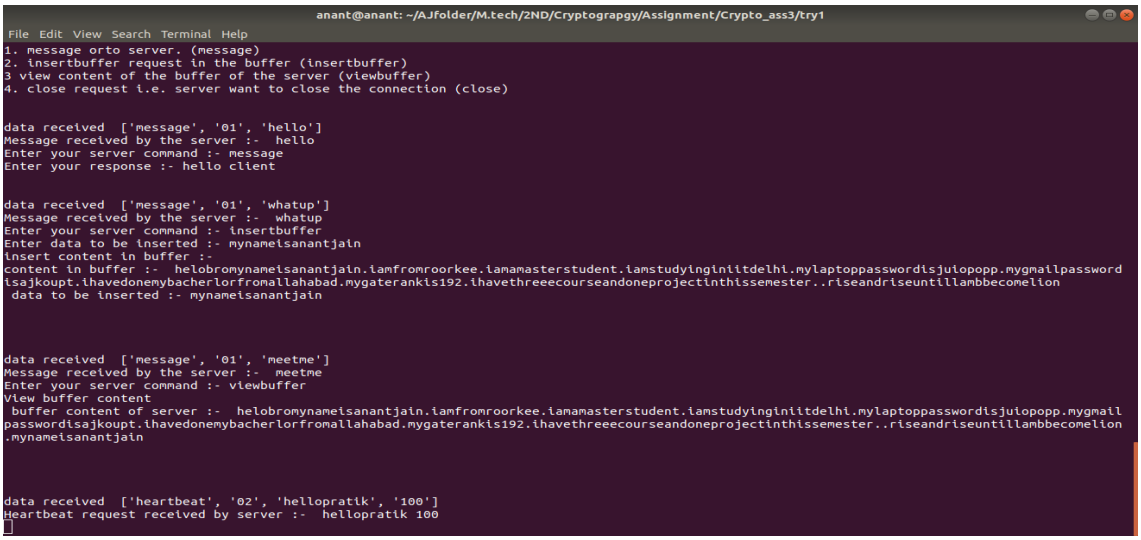

Figure 9: Server terminal after heartbeat request

4.6.6 Client Close command :-

Client sending close command

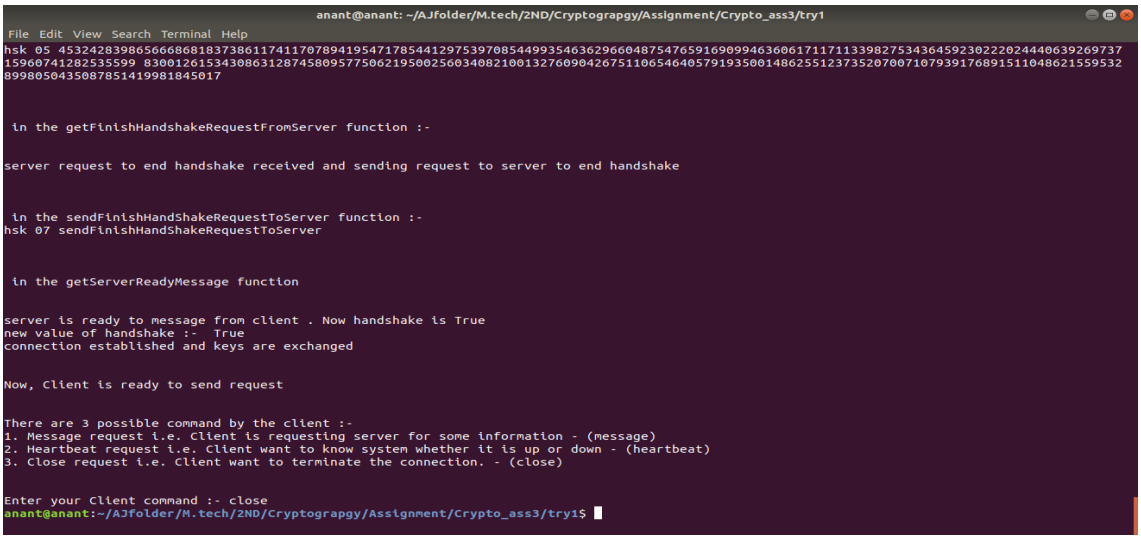

Figure 10: Client terminal after close command

Server after client close command :-

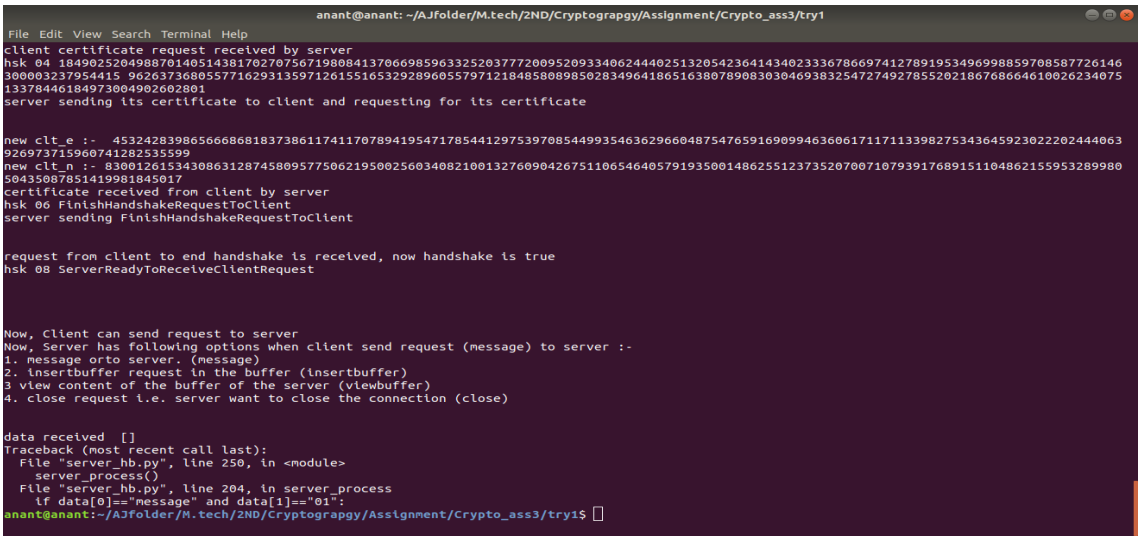

Figure 11: Server terminal after client close command

5 Refrences :-

- https://medium.com/@kasunpdh/ssl-handshake-explained-4dabb87cdce
- https://www.tutorialspoint.com/unix_sockets/ what_is_socket.html
- https://realpython.com/python-sockets/
- https://www.youtube.com/watch?v=hTK0pywfmDE
- https://heartbleed.com/