

Cryptography

Assignment3 Report

Vulnerabilities in standard cryptographic protocols

Anant Jain - (2019MCS2557)

August 14, 2020

Introduction

In this assignment , 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, BREACH 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 Explanation 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 got the name of heartbleed bug.

Heartbleed bug is a result of implementation fault in OpenSSL cryptography library, widely used in the TLS protocol. It occurs due to lack of valid check when heartbeat request is sent. The client sends 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 responds to client with the message of inputted heartbeat message length (it contains inputted heartbeat message and confidential information of server). By this, client gets the confidential data of server memory.

2.2 Diagram showing Heartbleed bug :-

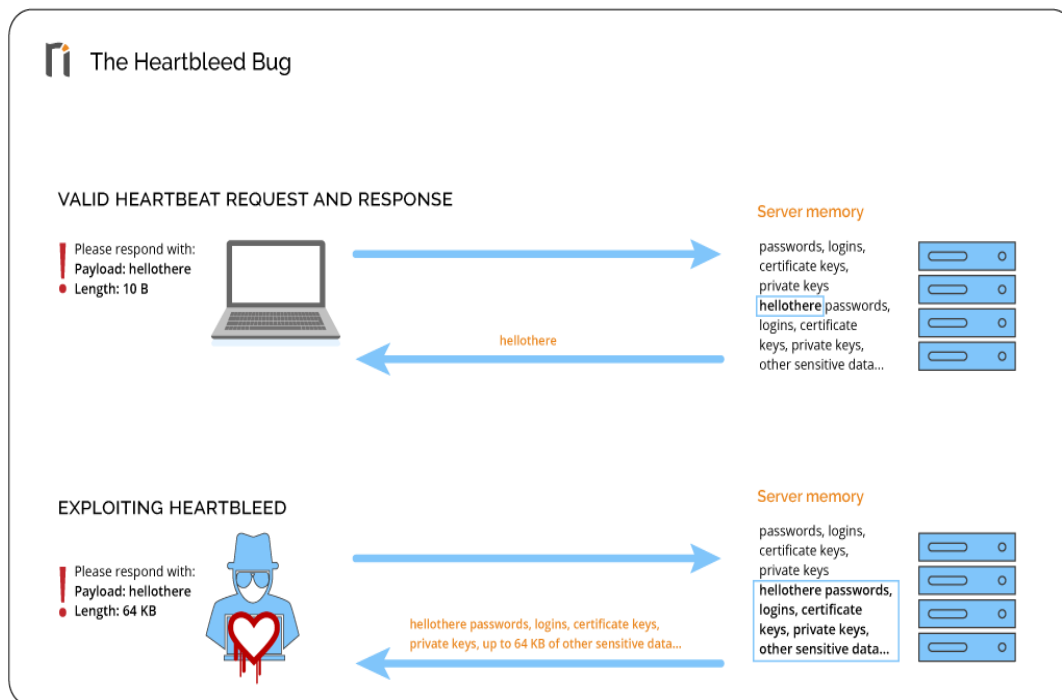


Figure 1: Diagram showing Heartbleed bug

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 Explanation 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 KeyGen-ForSerAndClt.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:
```

```

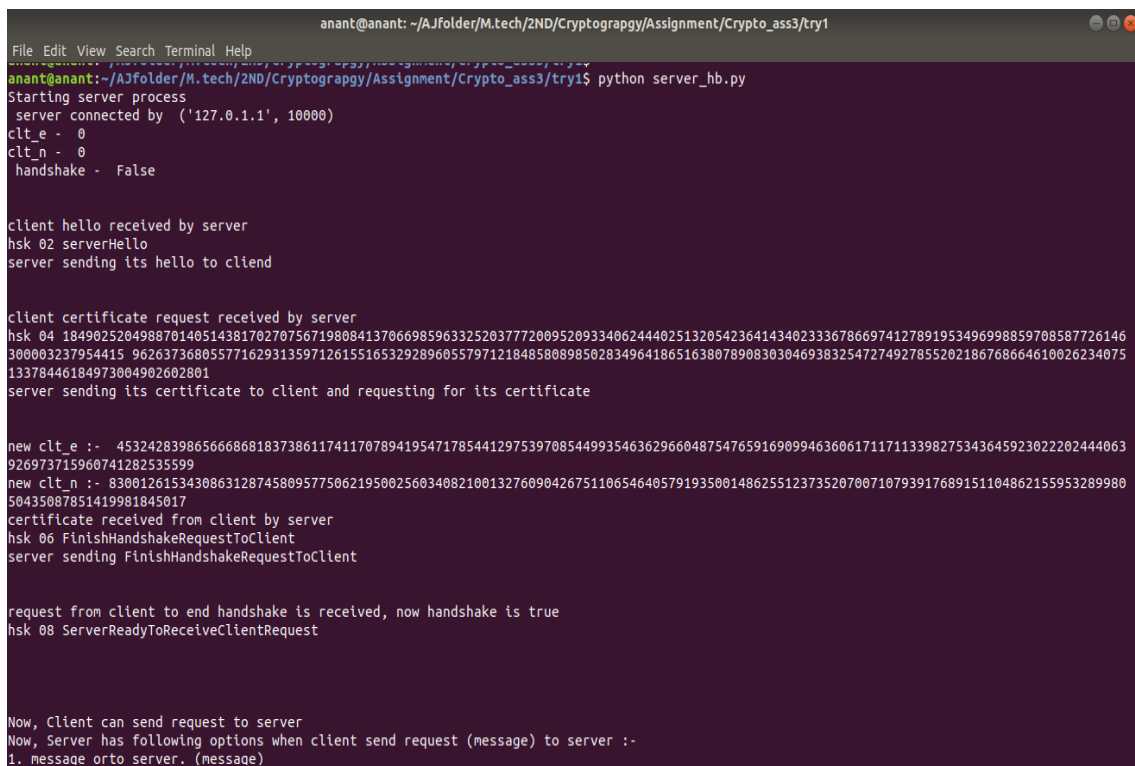
srhbmsg = client_socket.recv(1024)
srhbmsg = srhbmsg.decode()
if srhbmsg:
    print(' response of heartbeat message by server :- ',srhbmsg)
    print('\n')
    break

```

4.6 Screenshots of Terminal while running the code :-

4.6.1 Running Server file ,then running client file :-

Server side terminal :-



```

anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
anant@anant:~/AJfolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1$ python server_hb.py
Starting server process
server connected by ('127.0.1.1', 10000)
clt_e - 0
clt_n - 0
handshake - False

client hello received by server
hsk 02 serverHello
server sending its hello to cliend

client certificate request received by server
hsk 04 1849025204988701405143817027075671980841370669859633252037720095209334062444025132054236414340233367866974127891953496998859708587726146
300003237954415 96263736805577162931359712615516532928960557971218485808985028349641865163807890830304693832547274927855202186768664610026234075
13378446184973004902602801
server sending its certificate to client and requesting for its certificate

new clt_e :- 4532428398656668681837386117411707894195471785441297539708544993546362966048754765916909946360617117113398275343645923022202444063
926973715960741282535599
new clt_n :- 83001261534308631287458095775062195002560340821001327609042675110654640579193500148625512373520700710793917689151104862155953289980
50435087851419981845017
certificate received from client by server
hsk 06 FinishHandshakeRequestToClient
server sending FinishHandshakeRequestToClient

request from client to end handshake is received, now handshake is true
hsk 08 ServerReadyToReceiveClientRequest

Now, Client can send request to server
Now, Server has following options when client send request (message) to server :-
1. message orto server. (message)

```

Figure 2: Server terminal after handshaking

Client side terminal :-

```

anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
anant@anant:~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1$ python client_hb.py
Starting client process
ser_e - 0
ser_n - 0
handshake - False
----- handshaking started from client side -----
in the sendClientHelloToServer function
hsk 01 clientHello

client hello has been sent , now receive server hello

in the function getServerHello of client :-
data we get after decoding :- ['hsk', '02', 'serverHello']

server hello received , now send request for certificate

in the sendCertReqToServer function
hsk 03 CertificateRequestToServerByClient

in the function getServerCertificate of client :-

server certificate received , value -> 18490252049807014051438170270756719808413706690596332520377720095209334062444025132054236414340233367866
974127891953496998859788587726146300003237954415 96263736805577162931359712615516532928960557971218485800985028349641865163807890830364693832547
27492785520218676866461002623407513378446184973004902602801

```

Figure 3: Client terminal after handshaking

4.6.2 Message command from client and the from server :-

Client side terminal :-

```

anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help

in the sendFinishHandshakeRequestToServer function :-
hsk 07 sendFinishHandshakeRequestToServer

in the getServerReadyMessage function

server is ready to message from client . Now handshake is True
new value of handshake :- True
connection established and keys are exchanged

Now, Client is ready to send request

There are 3 possible command by the client :-
1. Message request i.e. Client is requesting server for some information - (message)
2. Heartbeat request i.e. client want to know system whether it is up or down - (heartbeat)
3. Close request i.e. Client want to terminate the connection. - (close)

Enter your Client command :- message
Wrong command send again

Enter your Client command :- message
Enter your message to server :- hello
clsng - message 01 hello
inputted message sent to server

response of server received - hello client

```

Figure 4: Client terminal after message command

Server side terminal :-


```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
server sending its hello to cliend

client certificate request received by server
hsk 04 18490252049887014051438170270756719808413706698596332520377720095209334062444025132054236414340233367866974127891953496998859708587726146
300003237954415 96263736805577162931359712615516532928960557971218485080895028349641865163807890830304693832547274927855202186768664610026234075
13378446184973004962602801
server sending its certificate to client and requesting for its certificate

new clt_e :- 4532428398656668681837386117411707894195471785441297539708544993546362966048754765916909946360617117113398275343645923022202444063
926973715960741282535599
new clt_n :- 83001261534308631287458095775062195002560340821001327609042675110654640579193500148625512373520700710793917689151104862155953289980
50435087851419981845017
certificate received from client by server
hsk 06 FinishHandshakeRequestToClient
server sending FinishHandshakeRequestToClient

request from client to end handshake is received, now handshake is true
hsk 08 ServerReadyToReceiveClientRequest

Now, Client can send request to server
Now, Server has following options when client send request (message) to server :-
1. message orto server. (message)
2. insertbuffer request in the buffer (insertbuffer)
3 view content of the buffer of the server (viewbuffer)
4. close request i.e. server want to close the connection (close)

data received ['message', '01', 'hello']
Message received by the server :- hello
Enter your server command :- message
Enter your response :- hello client
```

Figure 5: Client terminal after message command

4.6.3 Server Insert buffer command :-

```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
new clt_n :- 83001261534308631287458095775062195002560340821001327609042675110654640579193500148625512373520700710793917689151104862155953289980
50435087851419981845017
certificate received from client by server
hsk 06 FinishHandshakeRequestToClient
server sending FinishHandshakeRequestToClient

request from client to end handshake is received, now handshake is true
hsk 08 ServerReadyToReceiveClientRequest

Now, Client can send request to server
Now, Server has following options when client send request (message) to server :-
1. message orto server. (message)
2. insertbuffer request in the buffer (insertbuffer)
3 view content of the buffer of the server (viewbuffer)
4. close request i.e. server want to close the connection (close)

data received ['message', '01', 'hello']
Message received by the server :- hello
Enter your server command :- message
Enter your response :- hello client

data received ['message', '01', 'whatup']
Message received by the server :- whatup
Enter your server command :- insertbuffer
Enter data to be inserted :- mynameisanantjain
Insert content in buffer :-
content in buffer :- helobromynamesisanantjain.lamfromroorkee.lamasterstudent.lamstudyinginitdelhi.mylaptoppasswordisjuolopp.mygmailpassword
isajkaupt.lhavedonemybachelorlorfromallahabad.mygaterankis192.lhavethreecourseandoneprojectinthissenester..riseandriseuntillambbecomellon
data to be inserted :- mynameisanantjain
```

Figure 6: Server terminal after insertbuffer command

4.6.4 Server view buffer command :-

```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help

Now, Client can send request to server
Now, Server has following options when client send request (message) to server :-
1. message orto server. (message)
2. insertbuffer request in the buffer (insertbuffer)
3 view content of the buffer of the server (viewbuffer)
4. close request i.e. server want to close the connection (close)

data received ['message', '01', 'hello']
Message received by the server :- hello
Enter your server command :- message
Enter your response :- hello client

data received ['message', '01', 'whatup']
Message received by the server :- whatup
Enter your server command :- insertbuffer
Insert content in buffer :- mynameisanantjain
content in buffer :- helobronymynameisanantjain.iamfromroorkee.ianamasterstudent.ianstudyinginiitdelhi.mylaptoppasswordisjuiopopp.mygmailpassword
isajkoupt.ihavedonemybacherlorfromallahabad.mygaterankis192.ihavethreecourseandoneprojectinthissemester..riseandriseuntillambbecomelion
data to be inserted :- mynameisanantjain

data received ['message', '01', 'meetme']
Message received by the server :- meetme
Enter your server command :- viewbuffer
View buffer content
buffer content of server :- helobronymynameisanantjain.iamfromroorkee.ianamasterstudent.ianstudyinginiitdelhi.mylaptoppasswordisjuiopopp.mygmail
passwordisajkoupt.ihavedonemybacherlorfromallahabad.mygaterankis192.ihavethreecourseandoneprojectinthissemester..riseandriseuntillambbecomelion
.mynameisanantjain
```

Figure 7: Server terminal after view command

4.6.5 Heartbleed vulnerability simulation :-

Client sending heartbeat request and getting its response:-

```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help

response of server received - hello client

Enter your Client command :- message
Enter your message to server :- whatup
clnsg - message 01 whatup
inputted message sent to server

response of server received - your request has received

Enter your Client command :- message
Enter your message to server :- meetme
clnsg - message 01 meetme
inputted message sent to server

response of server received - your request has received

Enter your Client command :- heartbeat
Enter your heartbeat message :- hellopratik
Enter heartbeat message length :- 100
inputted heartbeat message is sent to sever.

response of heartbeat message by server :- hellopratikhelobronymynameisanantjain.iamfromroorkee.ianamasterstudent.ianstudyinginiitdelhi.mylaptop
p

Enter your Client command :-
```

Figure 8: Client terminal after heartbeat request

Server responding to heartbeat request of Client :-

```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
1. message orto server. (message)
2. insertbuffer request in the buffer (insertbuffer)
3 view content of the buffer of the server (viewbuffer)
4. close request i.e. server want to close the connection (close)

data received ['message', '01', 'hello']
Message received by the server :- hello
Enter your server command :- message
Enter your response :- hello client

data received ['message', '01', 'whatup']
Message received by the server :- whatup
Enter your server command :- insertbuffer
Enter data to be inserted :- mynameisanantjain
insert content in buffer :-
content in buffer :- helobromynamesanantjain.lamfromroorkee.lanamasterstudent.lamstudyinginitdelhi.mylaptoppasswordisjuiopopp.mygmilpassword
isajkaupt.lhavedonemybacherlorfromallahabad.mygaterankis192.lhavethreecourseandoneprojectinthissemester..riseandriseuntillambbecomelion
data to be inserted :- mynameisanantjain

data received ['message', '01', 'meetne']
Message received by the server :- meetne
Enter your server command :- viewbuffer
View buffer content
buffer content of server :- helobromynamesanantjain.lamfromroorkee.lanamasterstudent.lamstudyinginitdelhi.mylaptoppasswordisjuiopopp.mygmil
passwordisajkaupt.lhavedonemybacherlorfromallahabad.mygaterankis192.lhavethreecourseandoneprojectinthissemester..riseandriseuntillambbecomelion
.mynameisanantjain

data received ['heartbeat', '02', 'hellopratik', '100']
Heartbeat request received by server :- hellopratik 100
```

Figure 9: Server terminal after heartbeat request

4.6.6 Client Close command :-

Client sending close command

```
anant@anant: ~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
hsk 05 4532428398656686818373861174117078941954717854412975397085449935463629660487547659169099463606171171133982753436459236222624440639269737
15960741282535599 830012615343086312874580957750621950025603408210013276090426751106546405791935001486255123735207007107939176891511048621559532
8998050435087851419981845017

in the getFinishHandshakeRequestFromServer function :-
server request to end handshake received and sending request to server to end handshake

in the sendFinishHandShakeRequestToServer function :-
hsk 07 sendFinishHandShakeRequestToServer

in the getServerReadyMessage function
server is ready to message from client . Now handshake is True
new value of handshake :- True
connection established and keys are exchanged

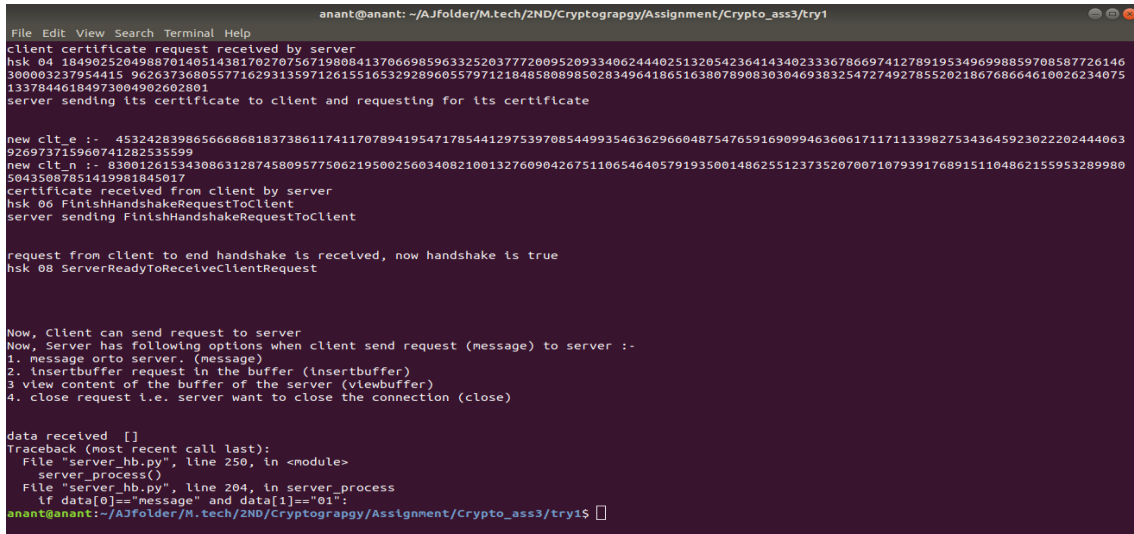
Now, Client is ready to send request

There are 3 possible command by the client :-
1. Message request i.e. Client is requesting server for some information - (message)
2. Heartbeat request i.e. Client want to know system whether it is up or down - (heartbeat)
3. Close request i.e. Client want to terminate the connection. - (close)

Enter your Client command :- close
anant@anant:~/AJfolder/M.tech/2ND/Cryptograpgy/Assignment/Crypto_ass3/try1$
```

Figure 10: Client terminal after close command

Server after client close command :-



```
anant@anant: ~/AJFolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1
File Edit View Search Terminal Help
client certificate request received by server
hsk 04 18490252049887014051438170270756719808413706698596332520377720095209334062444025132054236414340233367866974127891953496998859708587726146
300003237954415 96263736805577162931359712615516532928960557971218485808985028349641865163807890830304693832547274927855202186768664610026234075
13378446184973004902602801
server sending its certificate to client and requesting for its certificate

new clt_e :- 4532428398656668681837386117411707894195471785441297539708544993546362966048754765916909946360617117113398275343645923022202444063
926973715960741282535599
new clt_n :- 83001261534308631287458095775062195002560340821001327609042675110654640579193500148625512373520700710793917689151104862155953289980
50435007051419901045017
certificate received from client by server
hsk 06 FinishHandshakeRequestToClient
server sending FinishHandshakeRequestToClient

request from client to end handshake is received, now handshake is true
hsk 08 ServerReadyToReceiveClientRequest

Now, Client can send request to server
Now, Server has following options when client send request (message) to server :-
1. message orto server. (message)
2. insertbuffer request in the buffer (Insertbuffer)
3 view content of the buffer of the server (viewbuffer)
4. close request i.e. server want to close the connection (close)

data received []
Traceback (most recent call last):
  File "server_hb.py", line 250, in <module>
    server_process()
  File "server_hb.py", line 204, in server_process
    if data[0]=="message" and data[1]=="01":
anant@anant:~/AJFolder/M.tech/2ND/Cryptograpy/Assignment/Crypto_ass3/try1$
```

Figure 11: Server terminal after client close command

5 References :-

- <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/>