SECURE CHAT ROOM USING SOCKET PROGRAMMING

Computer Networks & Security Lab

Submitted by:

Shubhay Gautam (21104051)

Mogish Hashmi (21104038)

Avijot Singh (21104036)

SUBMITTED TO : **Dr. Janardan Verma**

Department of CSE & IT,

Jaypee Institute of Information Technology, Noida

November 2023

Table of Contents

INTRODUCTION

Chatting Applications are very popular in today's world. Chat applications offer communication services free of charge, and the majority of them are free to install, which makes them very appealing popular with customers. These applications offer different services and built-in features to their users.

Millions of smartphone users use chat applications on a daily basis. According to a report in 2014, there were 2.5 billion mobile broadband subscriptions worldwide whereas there are about 7.75 billion mobile broadband subscriptions in the year 2020[1]. This growth and accessibility to an Internet connection has brought opportunities for many Instant Messaging applications to enter a new area of communication.

Chat is to participate in a synchronous text, video or audio exchange with one or more people over a computer network. There is the need to ensure confidentiality of communication to breed honest and frank chatting free from fear of eavesdropping and breach of privacy.A Secure Chat System is a system which improves communication between two or more people over the internet in a way that seriously attempts to be free from risk of involvement of an unauthorized persons.

There are a number of chat systems available like Google Talk, Skype, Facebook, etc. These 'free' chat system providers use clients' information for marketing and may sell it to buyers who need them. Possession of chat messages is also lost to the provider. Thus, having a private chat system will reduce all these confidential, privacy, and possession security risks.

PROBLEM STATEMENT

Malicious users are always interested in hacking servers and revealing information about users which sometimes leads to a great loss of information or might be used to blackmail someone. Unfortunately, instant messaging applications are also not an exception. There are many mobile chat applications available for users that claim to be providing confidentiality, integrity and security of user's information. However, daily hacking news proves that many developers do not consider security as the primary goal of their applications and many agents take advantage of it.

For example, while many chat applications are free to use, they equip the application with built-in processes which track every single movement. They must be careful with what is going to be published on social networks. The hackers store large amounts of information about the activities we do, the places we visit, the people with whom we interact, our hobbies, the food we like, etc. All this information can be used by an attacker to know our profile or plan and launch custom attacks. The information collected can be used even for kidnappings or extortions**.**

Thus, the aim of this project is to design a secure chat application which protects user's privacy.

IMPLEMENTATION & TECHNIQUES USED

Socket Programming

Sockets are bi-directional communication channels that connect a server to one or more clients. We create a socket on either end and use it to let a client communicate with other clients through the server. On the server side, the socket is associated with a server-side hardware port. Any client with a socket on the same port as the server socket can talk with it.

Multi-Threading

A thread is a sub-process that executes a series of commands for each other thread. As a result, each time a user connects to the server, a new thread is established for that user, and communication between the server and the client occurs through distinct threads based on socket objects produced for the sake of each client's identification.To set up this chat room, we'll need two scripts. One to keep the server up and running, and another for each client to connect to the server.

Server Side Script

The server-side script will attempt to create a socket and bind it to the user-specified IP address and port (windows users might have to make an exception for the specified port number in their firewall settings, or can rather use a port that is already open). After then, the script will remain open and accept connection requests, appending respective socket objects to a list to maintain track of active connections. Each time a user joins, a new thread is started for them. The server listens for a message in each thread and then sends it to other users that are currently on chat. If the server finds an issue while attempting to receive a message from a certain thread, that thread will be closed. Usage: On a local area network, this server can be set up by designating any computer as a server node and using that device's private IP address as the server IP address.If a local area network has a set of private IP addresses spanning from 192.168.1.2 to 192.168.1.100, for example, any computer on the network can operate as a server, and the other nodes can connect to the server node using the server's private IP address. It's important to pick a port that

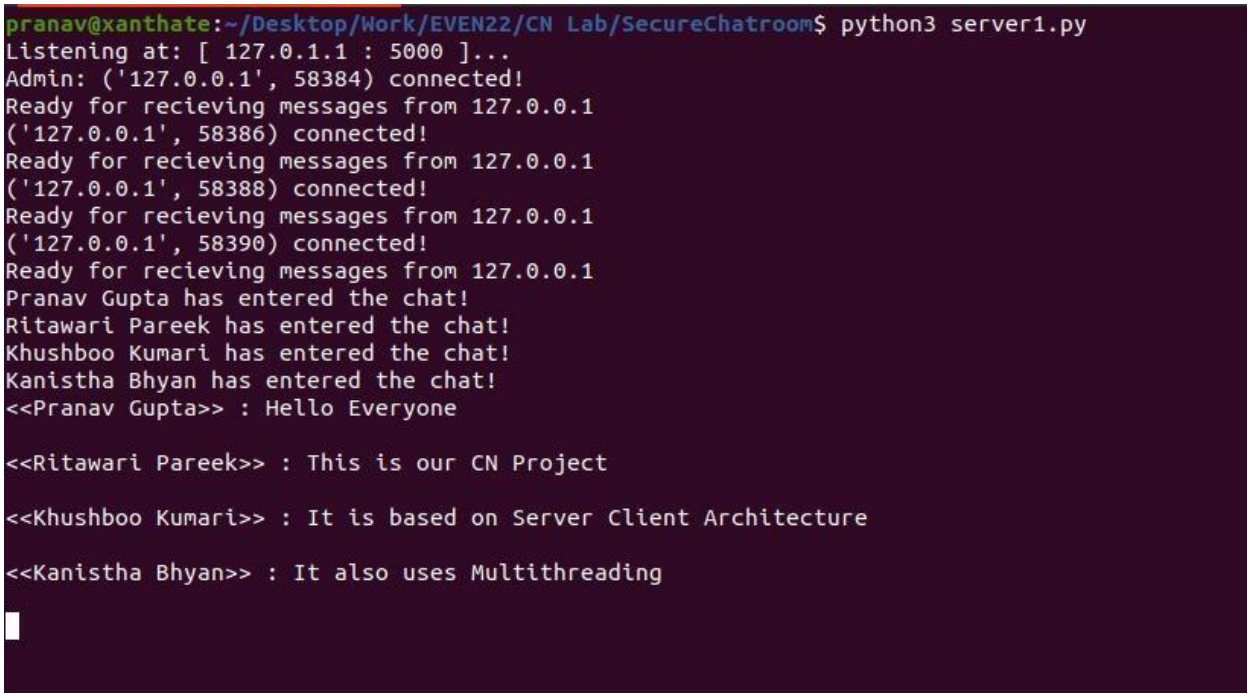
isn't currently in use. For example, ssh uses port 22 by default, whereas HTTP protocols use port 80 by default. As a result, these two ports should ideally not be used or modified.

Client-Side Script

The client-side script will simply try to connect to the server socket generated at the IP address and port provided. Once it's connected, it'll constantly check whether the input is coming from the server or the client, and redirect output accordingly. If the message comes from the server, it is shown on the terminal. If the input comes from the user, it sends the message to the server, where it will be disseminated to other users.

Screenshots

Server side



Client Side

```
pranav@xanthate:~/Desktop/Work/EVEN22/CN Lab/SecureChatroom$ python3 client1.py
Trying to connect to [ 127.0.1.1 : 5000 ]...
Connected to host...



        ##############################################################################################

                                    JIIT CHATROOM - CN LAB
                                      Connecting People

        ##############################################################################################


Enter your Enroll : 19803021
Enter your Password :
19803021 : Pranav Gupta

All set, type 'QUIT' to leave...
Ritawari Pareek has entered the chat!

Khushboo Kumari has entered the chat!

Kanistha Bhyan has entered the chat!

<<You>> : Hello Everyone
<<Ritawari Pareek>> : This is our CN Project


<<Khushboo Kumari>> : It is based on Server Client Architecture


<<Kanistha Bhyan>> : It also uses Multithreading


<<You>> :
```

Quit using keyword

```


<<Khushboo Kumari>> : It is based on Server Client Architecture


<<Kanistha Bhyan>> : It also uses Multithreading


<<You>> : quit
Quitting chat...

pranav@xanthate:~/Desktop/Work/EVEN22/CN Lab/SecureChatroom$
```

Authentication. Only jiit students allowed and that too with a valid password

```
pranav@xanthate:~/Desktop/Work/EVEN22/CN Lab/SecureChatroom$ python3 client1.py
Trying to connect to [ 127.0.1.1 : 5000 ]...
Connected to host...



       ##############################################################################################

                                  JIIT CHATROOM - CN LAB
                                    Connecting People

       ##############################################################################################


Enter your Enroll : 19803021
Enter your Password :

Invalid Password
--------------------Access Denied -----------------

-----------Invalid Username or Password -----------------

Enter your Enroll :
```

Secure Database

```
pranav@xanthate:~/Desktop/Work/EVEN22/CN Lab/SecureChatroom$ cat db.txt
19803001        Aakash  Agrawal 19513fdc9da4fb72a4a05eb66917548d3c90ff94d5419e1f2363eea89dfee1dd
19803002        Naivedya        Khare   1be0222750aaf3889ab95b5d593ba12e4ff1046474702d6b4779f4b527305b23
19803003        Khushboo        Kumari  2538f153f36161c45c3c90afaa3f9ccc5b0fa5554c7c582efe67193abb2d5202
19803004        Mehul   Porwal  db514f5b3285acaa1ad28290f5fefc38f2761a1f297b1d24f8129dd64638825d
19803005        Dharmesh        Malav   8180d5783fea9f86479e748f6d2d1196c4a8e143643119398c16367d2c3d50f2
19803006        Priyanshi       Raman   75f9793a7639faa0b83a2707d60cb21c42fe9f50992fc692390a26ac2a21b29e
19803007        Gautam  Maheshwari      5bfdfaaf7e1b1244192a1ede1ea70f09f5642190821c0005669a9afbca2dee2e
19803008        Kanistha        Bhyan   2ced6e7160a9e2cb4be29c200852bfc4fe29d7531ff3ffc51fc1407399d8d8b8
19803009        Ritawari        Pareek  b949a64fd5484e69191efb60d83f7f79195eeb2911c3eb5850af160841211f18
19803011        Kanishka        Khullar c0a09d876279cea6c57b4453c56737fd1b0c6c95e80b0a08ac48bcc97e719afd
19803012        Aniket  Kumar   542cbab799aabae8c7b3cd571e6c73395515ebd86044358cc3603d8e965881e0
19803013        Shagun  Sharma  f3c16dc3ef3ba55671b0ac2938730a4afc3867cf4c01ae9a09cfe4e2367666bd
19803014        Utkarsh Choudhary       94160c710f5c02b0deaa4f8f4f296471b19e1152390f31b545d75a3648e3332a
19803015        Agnideep        Mukherjee       561f1620b13d82f635b00720379773d89bf42945d257b39172941f5b63074dc3
19803016        Priyanshu       Jaiswal f16171f1256ff454c66ff516f9e61c69b2454f8065de099dd147452486b91211
19803018        Muskan  Sharma  9630a358e80519239f62f32ddd4e47f4e798dccc2a70c50841253ee54ead3d56
19803019        Garvita Nagpal  40c65d8f21b62fceddeaaadf221c5549f4fd230a214b80236584aa5f6b8ab357
19803020        Aman    Bhadouria       e6e90dc078b603b8a7f4788a464afea9762108073cfb40871c8861bfec714ed7
19803021        Pranav  Gupta   e2832f0c57bf6c819cbd084dbb237c498b01909afed4ecba20b00e0a8139d3d1
19803022        Parish  Bindal  69e44c3b1c786801d098ad5e40b9d347d300712254ea0e95d374620a08372f8f
19803023        Sanskar Gupta   4f184fcd08ea9259d231d74dedfe990b534c2ee568ca70bf09a7d6945d46bdff
19803024        Nikhil  Paleti  7b98cd8482ae01ac3aceb273c9e354d48c5660d98a2b723141f8a9989a546b6f
19803025        Divyanshu       Tiwari  d55c0a75cd34955bced4210f203dfe77f5b296892c2e786c091325d329c2482e
19803026        Divyansh        Chauhan 76475b1d780d2f58f852243140d9977e8b8c4ea81979e04afbf05fb3f0789a45pranav@xanthate:~/
ecureChatroom$
```

CONCLUSION

Thus we successfully built a server client chatting room withthe help of socket programming and multi threading which is specifically for jiit students. This is comparatively much secure as compared to traditional client server chatroom.

REFERENCES

1. https://datareportal.com/reports/digital-2020-global-digital-overview
2. https://www.tutorialspoint.com/socket.io/socket.io_chat_application.htm
3. https://ieeexplore.ieee.org/document/6394395
4. https://ieeexplore.ieee.org/abstract/document/8524573
5. https://link.springer.com/chapter/10.1007/978-1-4842-7138-4_4