Threat Modelling - Client-Server Live Chat Service

Akshay Kaikottil
Johns Hopkins University, Information security Institue
Baltimore, MD
akaikot1@jh.edu

***Abstract*—Chatroom applications are commonly used in today's world to send messages to a targeted group of users. Most chat applications make use of sockets to enable communication in real time. This report identifies the architecture of a simple open-source live chat service and analyzes the possible threats associated with its different components which is necessary to ensure that the present version complies with necessary security requirements.**

Keywords—chatroom, sockets, open-source

I. Introduction

A chatroom is an application that is primarily used to describe any form of synchronous conferencing, occasionally even asynchronous conferencing. Many people use chatroom s to communicate messages to a group of people. Attackers can target these applications to breach the security of the application and access their private data. Hence, it became our primary motivation to assess the threat scenarios related to an open-source live chat application. Therefore, we do threat modelling for imminent insecurities on our chosen open-source chatroom application – Server-Client Live Chat Service.

II. Architecture Overview

Server-Client Live Chat Service makes use of sockets to send and receive data over a network. The application has two major components; a sever and a client. The chatroom is initiated by the admin by calling the server function. The users then join this server as clients by providing the server address, username and password provided by the admin.

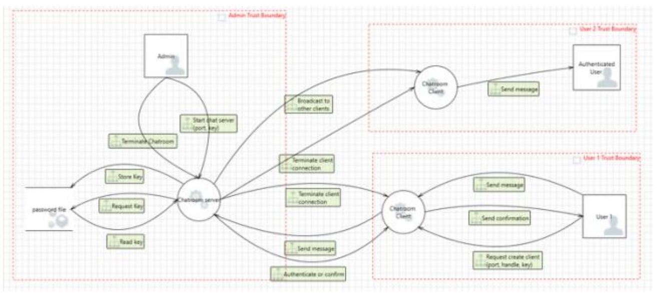
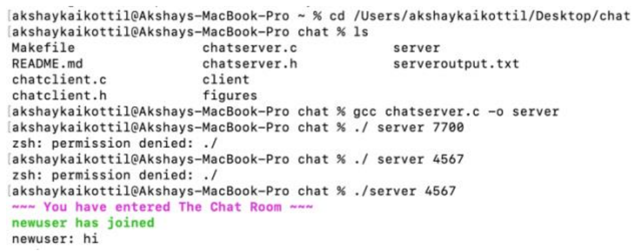


Figure 2.1: Threat modelling diagram for Server-Client Live Chat Service

III. Application Analysis

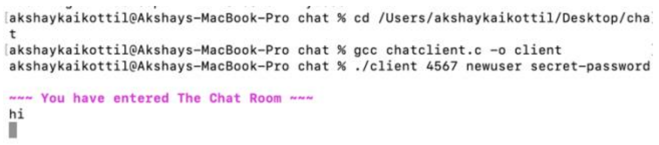
To understand the working and connection flow of the Op Server-Client Live Chat Service, we took the binaries and compiled them and installed the Server-Client Live Chat Service as shown in the Figure below:

To understand the working and connection flow of the Op Server-Client Live Chat Service, we took the binaries and compiled them and installed the Server-Client Live Chat Service as shown in the figures below:



```
akshaykaikottil@Akshays-MacBook-Pro ~ % cd /Users/akshaykaikottil/Desktop/chat
akshaykaikottil@Akshays-MacBook-Pro chat % ls
Makefile                chatserver.c            server
README.md               chatserver.h            serveroutput.txt
chatclient.c            client
chatclient.h            figures
akshaykaikottil@Akshays-MacBook-Pro chat % gcc chatserver.c -o server
akshaykaikottil@Akshays-MacBook-Pro chat % ./ server 7700
zsh: permission denied: ./
akshaykaikottil@Akshays-MacBook-Pro chat % ./ server 4567
zsh: permission denied: ./
akshaykaikottil@Akshays-MacBook-Pro chat % ./server 4567
~~~ You have entered The Chat Room ~~~
newuser has joined
newuser: hi
```

Figure 3.1: Compilation and execution of server binary.



```
akshaykaikottil@Akshays-MacBook-Pro chat % cd /Users/akshaykaikottil/Desktop/cha
t
akshaykaikottil@Akshays-MacBook-Pro chat % gcc chatclient.c -o client
akshaykaikottil@Akshays-MacBook-Pro chat % ./client 4567 newuser secret-password

~~~ You have entered The Chat Room ~~~
hi
```

Figure 3.2: Compilation and execution of client binary.

IV. Project Execution

To investigate the security issues, we first begin by relating to the documents and code available in the project repository. We focus on the execution of the chat application on MacOS and test the possible imminent threats. Upon analysis of the various function in the application, we create a threat model to get a detailed analysis of the architecture used in Server-Client Live Chat Service. We decompose the application further to identify, document and rate the threats concerning each attribute. In doing so, more useful details unveil, and we gain an understanding of the overall Security of the System.

Threat Modeling Process:

1. Firstly, we identified the critical assets that the functional system must protect.
2. We used Microsoft threat modeling tool to document the architecture of Server-Client Live Chat Service, including subsystems, related subsidiaries, trust boundaries and the organized data flow.
3. Then we decompose the architecture further, to include the underlying network, protocols used, and the host environment design, to create a security profile for Server-Client Live Chat Service. The aim of this profile was to uncover and identify vulnerabilities in the design, implementation, and

deployment configuration present in Server-Client Live Chat Service.

4. We now identify the threats that could be associated to the application, with respect to an attacker and his possible outplay. The knowledge of the architecture and potential vulnerabilities of the ap- plication will help us in this process.
5. We now document each threat using an available threat template that defines important attributes to be captured for each identified threat.
6. We then rate these threats from most significant to least significant as per priority. The most significant threats present the most severe risk, and the rating process maps the probability of the threat against damage that could take place and hence this provides a more accurate standard.

V. Projet Observation

Server-Client Live Chat Service makes use of web sockets to facilitate connections between multiple clients. The application fails to make use of common security practices and hence is vulnerable to even the most common attacks. We have identified two major trust boundaries, the server, and the clients. It is to be noted that data flow between trust boundaries is not sent over a secured channel nor is it being encrypted. By threat modeling we were able to identify all the possible threats including spoofing, parameter manipulation, privilege escalation and the most significant attack pattern possible for such a simple system.

By rating the threats, we discovered with Server-Client Live Chat Service, we identified that one of the high-risk threats is gaining privilege access to the server by insertion of malicious code. An attacker can sniff the key used to connect to the server and then send malicious data which is not validated on the server.

VI. Conclusion

We were able to successfully identify the threats associated with the selected target using Microsoft threat modelling tool. Also, we were able to classify the threats associated to Server-Client Live Chat Service and rate them based on severity. These threats can be further generalized and can be associated to other chat applications. All these products can also be susceptible to the same attack patterns due to the similar functionalities between them.

Acknowledgment

The author wishes to thank Reuben Johnston for his support for the successful completion of the Threat Modelling on the Server-Client Live Chat Service using Microsoft Threat Modelling tool.

References

[1] Server-Client Live Chat Service, https://github.com/andrea-covre/client-server-live-chat-service

[2] Cybersecurity Chat: The Security Risks Of Popular Collaboration Apps, https://www.forbes.com/sites/forbestechcouncil/2019/04/03/cybersecurity-chat-the-security-risks-of-popular-collaboration-apps/?sh=506199493e1d

[3] The Instant Messaging Menace: Security Problems in the Enterprise and Some Solutions, https://www.sans.org/white-papers/479/

Key terms

Threat: Threat is a possible risk that might exploit a vulnerability to breach security.

Vulnerability: A flaw in a system that leaves information expose to a threat.

Attack or Exploit: Code or data which takes advantage of a vulnerability in a system for malicious intent.