Elliott Picker
Secure Chat
MSCS_630L_711_20S

Secure Chat is an open-source lightweight desktop app that enables encrypted end-to-end communication through a centralized server. All encryption occurs on the client side so the messages broadcasted to and from the server are unreadable by anyone other than the Secure Chat users with the appropriate key. The symmetric encryption depends on the users sharing the key external to the application such that the server plays no part in identifying or authenticating any users.

Abstract
In today's world it seems that cyber hacking and data breaches are a recurring theme in the news. Properly securing all information including communication is a challenging task that can have severe consequences when not done properly. While some cases can lead to embarrassing conversations making it to the public, business confidential secrets can also be leaked and in some cases even compromise national safety. End-to-end encryption offers its own challenges and often leaves possible exposures where data may not be fully secured.

Introduction
Secure Chat is a lightweight open-source desktop app that enables end-to-end encrypted communication through a centralized server. All encryption occurs on the client side so the messages broadcasted to and from the server are unreadable by anyone other than the Secure Chat users with the appropriate key. Because the program is very lightweight, any skeptics of the security of the program can easily read through the source code and ensure that there are no backdoors present and all content will be properly safeguarded. Because the server has no role in identifying or authenticating users, the server cannot be spoofed or otherwise hacked into allowing any untrusted access to encrypted content.

Background and/or Related Work
While many existing end-to-end encrypted chat services exist today many people have doubts about the security and integrity of such services. For example, while the popular messaging service WhatsApp boasts end-to-end encryption, some people suspect possible “backdoors” which could allow for someone other than the intended end users to possibly access this data. While many government agencies are in favor of these types of backdoors for authorized use in necessary circumstances, once such backdoors exist, they become a prime candidate for hackers and other malicious users attempting to gain unauthorized access to the information. Additionally because many services allow a user to logon from multiple devices and access their same message histories, the server must somehow store the private key, and is tasked with authenticating users prior to allowing use of this key. If someone can falsely convince the server they are an authorized user, they can gain access to the appropriate secret key and access the encrypted messages.

Methodology
A key file created from the Secure Chat program is vital to the integrity of the encrypted data. The Key Creator program converts a unique passphrase into a bit encryption key. The key file consists of both this unique key as well as the path of the server. If possible, this key file can be distributed physically or through some other protected means, to ensure only the authorized users have the appropriate key. If the key file cannot be securely shared, a passphrase and ip address of the server can be shared, allowing each user to create their own identical key file prior to participating in the secure chat session. There is no use of public keys in the key sharing as the server takes no part in deciding who users are or who is

authorized to the chat. Authorization is done completely external to the Secure Chat application, assuming that only authorized users will have access to the appropriate key or passphrase used. While this may seem like a disadvantage in that another means of communication is necessary for key sharing, it enables the server to know nothing of the session key being used. This prevents the server from being able allow untrusted access, either through a valid source such as a government under a warrant or an invalid source such as a malicious hacker.

The Secure Chat client program encrypts all messages before sending them to the server through use of the javax.crypto.* libraries performing symetric encryption using the key found in the key file. The client program additionally decrypts all messages received from the server with this same key, so that the end user can read the messages in plain text. This assures that the authorized users of the chat client are the only people able to access the content of the messages exchanged. The server itself and the communication sockets are unable to decipher the messages being sent back and forth. Additionally while the server will be aware of the number of connected users, it has no knowledge of the identity of the users. Anyone with the valid key will be permitted to join the chat and use any username they wish. By using the javax.crypto libraries, secure chat is not tasked with implementing the encryption itself nor any possible bugs or vulnerabilities associated with that. As vulnerabilities are detected and fixed in the javax.crypto library, Secure Chat can be recompiled to include those changes.

Experiments

Discussion and/or Analysis

Conclusion

Secure Chat is an easy to use solution for end-to-end encryption messaging. Because the keys are not passed through the application in any way, a properly protected and shared key file will ensure the security of messages transmitted through the application. Additionally since the program is lightweight and open-sourcem all of it's integrity claims can be easily validated by any skeptical users. Because users are free to recompile the program at any time, and host the chatServer themselves, users can be assured that they are in complete control of their conversations. For any users looking for a simple way to share a confidential chat with multiple users, Secure Chat is a simple solution where users can be assured their message content will only be accessible to those who have the key.

References / Bibliography

[1] Kenyon, Miles. “Secure Your Chats: Why Encrypted Messaging Matters.” *The Citizen Lab*, 8 May 2019, citizenlab.ca/2017/11/secure-your-chats-encrypted-messaging/.

[2] Thakkar, Jay. “End-to-End Encryption: The Good, the Bad and the Politics.” *Hashed Out by The SSL Store™*, 2 Dec. 2019, www.thesslstore.com/blog/end-to-end-encryption-the-good-the-bad-and-the-politics/.

[3] Strickland, Jonathan. "End to End Encryption Services." *Tech Stuff. HowStuffWorks, Inc*. 24 April, 2018 https://www.iheart.com/podcast/105-techstuff-26941194/episode/end-to-end-encryption-services-29237343/

Current Status: As of April 12, the chat client and server programs can be run to provide the chat service. At this time no encryption is done, and the location of the server is taken as an input argument to the main method of the chatClient class. While the chat functionality is handled in the Jframe GUI, the current program prompts for the users name through System.out and System.in

Next Steps: I will be adding some symmetric AES encrpytion calls to chatClient such that outbound messages are encrypted and inbound messages are decrypted with the private key found in the key file. Currently there is no notification to a user when another user logs off. While the server doesn't know the name of the user logging off, it will be able to notify users that some user has left the chat. Additionally I haven't yet coded the key creator program described which will take a passphrase and convert it to an 128 bit key and store that as well as the server ip address into the key file.