CECS 478 Project Phase I

Pied Piper

Version 1

1 Application Properties

The chat application will provide end-to-end encryption for messages sent through the service. The application will be implemented for Android through the Java programming language. We will be using a RESTful server using the LAMP stack for our backend. We will use OpenSSL to provide confidentiality and generate the keys. We will use Let's Encrypt to set up a free certificate .

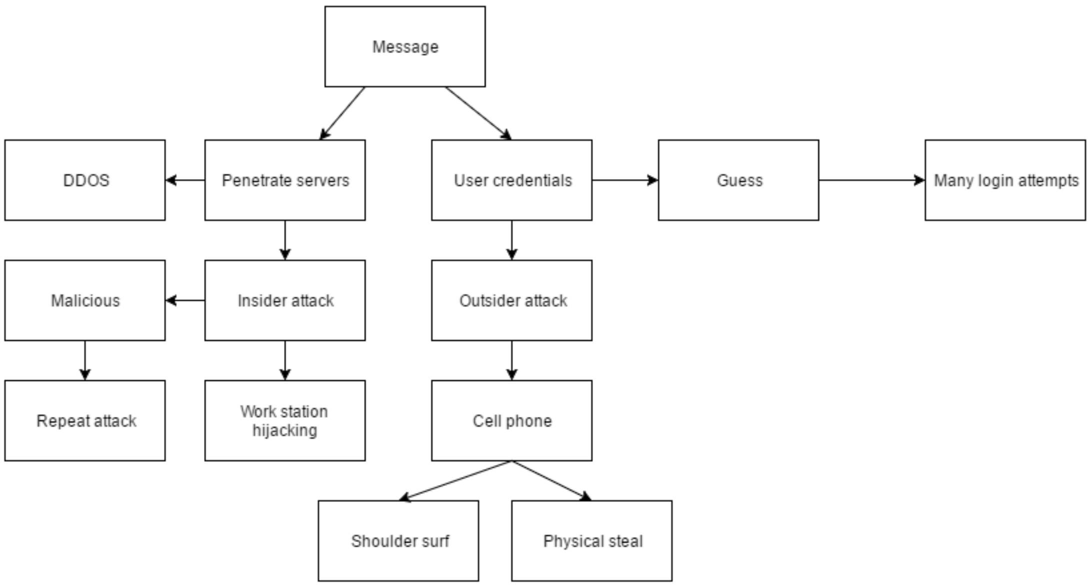
2 Assets/Stakeholders

For our application the stakeholder will be defined to be the user. The user is the most important and most valuable part of any application. Because of this the asset becomes the user's data. This data includes the message content they are sending as well as any other information they submit to the service; in this case their user name and password.

3 Adversarial Model

For our purposes we have two main adversaries. The insider and the outsider. To prevent an outsider attack (eavesdropping), we will implement Transport Layer Security (TLS). This will establish a secure connection between the client and the server. To prevent an insider attack (Man in the Middle), we will use OpenSSL to generate 2048 bit keys.

4 Possible Vulnerabilities

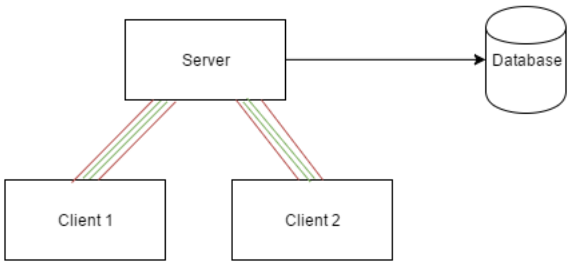


5 Related Work

Related works to this project include popular messaging applications such as WhatsApp, Signal, and Wickr.
https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf
https://www.wickr.com/security/how-it-works
https://open-whisper-systems.readme.io/

6 Solution

For our solution, we will implement the client side in Android Java for its access to many of the industry standard security protocols. Our client will connect to the server through HTTPS and TLS to establish a secure connection. The user will then register or log in with the service and their credentials will be stored by hashing to prevent them from being stolen easily. This establishes the so-called "red tunnel" of security. When the user wishes to add a friend, they will search for their username and send a request. The server will then communicate public keys created using OpenSSL through API calls. When a message is sent, the client will call the server to receive the public key and then send the message to the server. The message will live on the server until it is fetched by the receiving client.



7 Rigorous Analysis

Confidentiality will be achieved on the assumption that OpenSSL is a secure protocol for generating 2048 bit key pairs. Authentication will be achieved on the basis that user logins are stored securely and that user passwords meet modern requirements for a secure password. Integrity will be handled by TLS connections between the client and server to verify the server and prevent eavesdropping.