Names: Patrick White, Ryan Greaney, Anthony Arria, and Alex Wong

Date: 12/15/24

Professor Sashank

Sec.101

Milestone 1 - User Registration:

In this milestone, we implemented user registration using JSON files to help store the user data. JSON was chosen since it was easier to implement. Passwords are tested against complexity requirements, and then hashed using sha-256 with unique salts. The email is also hashed using a unique salt, and the name of the user is encrypted using AES in GCM mode, using the unsalted password hash as the key, which is not stored in any files.

Cybersecurity Aspects:

- Salted password hashes that help mitigate brute-force and dictionary attacks. The salts are generated using a battle tested random number generator, urandom in the os library, and sha-256 is a good hashing algorithm as it does not have any info leak, or collision.
- AES is a field tested encryption method and is almost guaranteed to safeguard against unauthorized third-party access, ex, if the filesystem is compromised

Milestone 2 - User login:

For user login, the major focus was to reuse code from the registration module to help in enhancing security measures. When prompted for a login, the user will enter their email and password, and the userfile will be read to obtain the salts, then the imputed data will be salted and hashed, then tested against the credentials on file.

Cybersecurity Aspect:

- Ensures secure authentication by not revealing any credential information

Milestone 3 - Adding Contacts:

For adding the contacts we use JSON files, similar to user data storage. Contacts are encrypted using the same method as the name from milestone 1, using the unsalted password hash as the key for AES in GCM mode.

Cybersecurity Aspect:

- Protects contact data with encryption techniques, ensuring confidentiality and integrity.
- Prevents unauthorized access to sensitive contact information.

Milestone 4 - Listing Contacts:

This was a very challenging milestone, particularly with trying to set-up the UDP communication with added security layers. We implemented TLS encryption wrapped around UDP to ensure secure data transmission, preventing man-in-the-middle attacks. We then added public and private key exchange to verify the identities of communicating entities. This is to help prevent someone impersonating the sender or receiver. Also we added a function to verify reciprocity in contact additions and the online status of contacts.One being the is_friend variable that guarantees when you added lets say **jane.doe@gmail.com** to your contact, she also added you to theirs. As well as an is_online function which helps ensure if the person is online or not so that you can send files to them. We have 2 variables in the Contact class that define if the contact is going to be displayed, and if they are verified as the real person. Contact.isfriend is only marked as True if the contact responds to the functions calls which use udp broadcast to obtain IP's for the email contacts, meaning they are online, and they also have you added as a contact. To ensure they are in fact the real person, the Contact.verified variable will only be set to true after the verify_addr function is called, which check the certificate of that IP associated with that contact. This is secure because the private keys are password protected ensuring that the only person who can use them is that person. These are password protected using the account password.

Cybersecurity Aspect:

- TLS over UDP prevents eavesdropping and ensures confidentiality
  During communication
- Public and Private key exchange enables mutual authentication and prevents impersonation
- Ensures secure contact visibility by verifying mutual addition and online status
- Password protected private keys ensure that even if a bad actor has obtained the filesystem of the machine, transmissions are still secure.

Milestone 5 - Secure File Transfer:

This milestone focused on securely transmitting large files over the network. To be able to achieve this, we used TLS sockets for secure communication and file transfers. The file is validated against the received file to ensure integrity before notifying the user of successful transfer. Error handling ensures when a file is correctly passed through or if something went wrong. Contacts are verified through a combination of SHA256 email hashing and public key checks using their provided CA's. The files are received over a TLS connection using hashed data chunks for integrity verification. The get_clientContext function manages user-specific certificate loading and key usage for TLS client connections.

Cybersecurity Aspect:

- Ensures secure file transmission through TLS encryption, which uses cutting edge solutions to possible attacks like timestamps, a mitigation against replay attacks.

Unfinished notes:

- This milestone went unfinished due to the long hours it took us to troubleshoot the code. We had a lot of issues with getting the y/n input to work, and it still currently does not, which does not allow the rest of the file transfer to work. However, our concept of doing it, although untested, we believe would work if we could get the program to iterate that part of the code. It is reading and writing the file in real time, using TLS as the file transfer method by hashing bytes in blocks, sending them, and waiting for either an ack, meaning the hash is good, or a hash-error meaning the file transfer failed and should be attempted again. We also had some issue making custom exceptions to raise when the file transfer did not work. We tested those in a separate file, and they worked just fine but keep erroring in our main project.