To: Instructor, CYBER-260-40

From: Dylan Forkey

Subject: Python project proposal

Date: Sept 18th, 2021

**Introduction**

For my project I am thinking of making a secure file transfer and storage application. The purpose of the application is to provide users with a reliable way to store their files as well as knowing that the files are encrypted not only during transfer but also once they are stored on the server.

For the final product, I am lacking pretty heavily when it comes to the actual user experience. Functionally, each part works, but only if each command is typed exactly, otherwise both the server and client disconnect. Additionally, the server does not continue to run after the client disconnects and when I attempted to restart the program automatically it broke the program because it attempted to reuse the same address and port that was previously used but because it is running so soon it says that the port and address are still in use. Additionally, a problem that I cannot seem to fix, is that sometimes the key exchange does work properly and so every once in a while, it appears as though the client program just breaks. Beyond those primary functional errors, the program works fine. The client has the ability to create an account or login, and the ability to upload/download files from the server. The key exchange allows for the files to be fully encrypted when they are stored on the file server. The client can also view the filenames that they have uploaded. The last service which is a bit of an addition is the port scanner which allows the client to enter an Ipv4 address and scan that address for open ports.

**Major features and solution overview**

So the major features that I can think of right now would be the user creation feature, the encryption feature, the networking feature, and the login ability. I am sure I will come up with one or two more “major” features over the course of the next couple of weeks but these are the ones I can think of now. The primary datatypes that I know I will be using constantly will be: Socket objects, Integers, Strings, arrays/lists/tuples, files, sqlite3 data.

I plan on utilizing object oriented programming for the sake of creating a consistent user connection that can be easily moved around without needing to pass 5 variables to each function that needs this information. As for the high-level solutions (Most of the instances below of the Socket sock value may be the passing of an object rather than Socket variable):

* User creation
  + I am thinking that I will use a SQLite database for creating users as this will allow for easy manipulation of the contents
  + This will include the following functions:
    - int create\_user(String username, String password)
      * Return 1 or 0 if user creation failed or succeeded
    - int login(String username, String password)
      * If login is successful will return 0 or 1 if failed
* Encryption
  + I will use the python pycryptodome (https://www.pycryptodome.org/en/latest/) library to create the keys for each user and these keys will be stored accordingly.
    - int encrypt(String data)
      * Return 1 or 0 to show success or failure
    - int decrypt(String data)
      * Return 1 or 0 to show success or failure
* Networking
  + I am just planning on using the python sockets library to implement my networking functions. I have prior experience working with this library so I plan to use it for this project
  + The networking functionality will need these functions at least:
    - int send\_file(String path\_to\_file, Socket sock)
      * Return 1 or 0 if success fails or succeeds
    - int receive\_file(Socket sock)
      * Return 1 or 0 if file is not received or received
* Login ability
  + Once users are created they have to be able to properly log in and maintain a session. I plan on using something like a session-key to allow for extended sessions which would allow for more expansion but due to time I may just create a single-session type client app where once the user is logged in they are logged in for as long as the application is open.
  + I already noted the needed login function but these functions may be needed as well
    - int send\_successful\_login(Socket sock)
      * Return value will be 1 or 0 depending on result
    - int send\_unsuccessful\_login(Socket sock)
      * Return value will be 1 or 0 to show successful execution of the function

**Data needed**

For this app, the only information needed is really information provided by the client. This would be information such as their real-name, their login id, their password, their encryption key, and any files that they hope to upload, or in the future download, and that is really about it for external data being provided to the system.

**Networking Functionality**

The primary elements of this project will use networking functionality. For the project I will use the TCP/IP protocol and the Python socket library to create a reliable connection between the client and the host. This connection will be maintained throughout the whole use of the application as it is required for logging in, downloading files, uploading files, creating an account, deleting files, etc. because for every action that the user wants to take there must be a way of getting that command to the server.

The primary data that will be sent over the socket connection will be files and general data. The files will be broken into sizeable chunks and sent with a clear beginning and ending message to inform the server when the whole file has been received. Additionally, general text data such as login information, filenames, commands. Because of the nature of the project, all of the networking functions are included in the major features section.

**Persistent Storage**

As for persistent storage, there is a sqlite3 database that holds all of the users information as well as the users symmetric key for encryption. There is also a set directory structure for storing client files on both the server and the client. It is assumed that the clients area all on the same machine for the sake of the project and so there is one set ./client/home directory that holds the clients folder to be uploaded or to store files that are downloaded from the server.

**Conclusion**

I will definitely end up refining the process as I begin creating it, but these are my first thoughts when it comes to the project creation. I believe the most difficulty will be found with implementing the end-to-end encryption, setting up the keys initially, and properly encrypting the files so they are whole files and not broken in any way during network transit. Although, when it comes to transferring files, python does make it pretty easy.