COMP.2300 - Intro to Computer Security

Course Project Milestone Security Details

Christian Kayego, Gavin Sadler, Adonis AlmonteRamirez

*Milestone 1*

User registration and login is handled using the python bcrypt library. This library allows for out-of-the-box simple credential management, allowing a user to simply encrypt a password, and check to see if a given password matches another’s encryption. Additionally, this library is used to generate salts. See DataManager.py for the implementation. Password hashes are stored alongside user emails in a file called userdata.json. See the screenshot below to see this.

*Milestone 2*

User login is handled by re-hashing the password with the given salt to see that it matches the stored password hash. If it does, then it allows the user to decrypt their user data. This is again handled by the bcrypt library.

*Milestone 3*

Adding contacts works by using the password taken at login to derive a key used by a Fernet encryptor which uses SHA256 as a backend. This implementation can be seen in the UserInstance class in the DataManager.py file. Essentially, the password is combined with the salt, hashed, and checked against the hashed password in the userdata.json file. If there is a match, then this same password is used to generate a key which is passed to a SHA256 Fernet encryption object. This object then is used to encrypt all user data, including contact information. In the screenshot below, this data is within the “data” field as part of the user object.

A screenshot of a computer

Description automatically generated

*Userdata.json, where the data field is encrypted using a key derived from the user’s password. Also shown is the salt and hashed passwords for users*

*Milestone 4*

Listing contacts, in this submitted version of the program, is not entirely complete. However, it is partially implemented as a secure TCP connection between clients.

*Milestone 5*

The only integrity check done when transferring files between clients is when a client sends a file, it also sends that files checksum calculated via a SHA256 hash. This is sent along with the file data. On the receiving end, the client receiving the file will also calculate the checksum of the file when it is finished being sent. If the checksums don’t match, then it will delete the data that was sent, as it is not the same data that the peer client intended to send. Either it was malformed while being sent or tempered with by a malicious actor. If the checksums match, then the file transferred successfully.