CSCI3403 - Project 2 Writeup

1) How did you hash the passwords in the file? Include information such as the hashing

algorithm, the library you used, how you appended the password with the salt before

hashing, and other information we’ve discussed related to password storage. Please

provide brief explanations as to why you chose to use what you did. Convince the reader

that your password storage algorithm is secure.

We used hashlib library to create hashes and used the uuid4 function which creates a random salt helping ensure privacy of the user’s password. Then we used the sha256 algorithm to add the hash and the salt to the password to generate an almost-unique, fixed size 256-bit (32-byte) hash.

2) Describe your approach to public/private key use. How did you generate your

public/private key pair? What library did you use to encrypt messages? Which folder did you store your keys? Please provide any details you think would be relevant.

We used ssh-keygen to create a secure public/private key pair. We used the RSA and PKCS1-OAEP library to encrypt messages. Stored the public and private keys in the client and server folders respectively.

3) How did you manage symmetric encryption? What encryption mode did you use, and why did you think that was best? What tradeoffs did you make for that encryption mode?

We instead used asymmetric encryption because the server used a private key to encrypt and decrypt messages while the client used a public key to encrypt and decrypt messages, as well. We thought it was the best to ensure the connection from the client to the server was unique and valid.

4) Provide a brief explanation of why your program would be secure from eavesdroppers if

you were to run it on a publicly visible network, from start to finish. Write your program

like you’re protecting yourself from Comcast or the NSA!

Our program makes diligent use of public and private keys for decryption and encryption. Whenever a message is passed between user-server or server-user it comes encoded such that anyone intercepting the message would not be able to read it without knowledge of the keys used.

5) Is your program secure from replay attacks? If not, why not? Use protocols that we

discussed in class to show that it isn’t vulnerable. If it is vulnerable, what could you do to

prevent it? Could you perform a replay attack against yourself?

Our program does seem susceptible to replay attacks, as the hash being sent back and forth is reused. In order to prevent this, we could implement functionality that would change the keys and encryption each time the message is transmitted from the source to the receiver.

6) What else did you learn from the project? Did you have to do some research?

Collectively, or individually, give some of the key takeaways you had when doing this.

We did have to do research to complete the project. Although it was not necessarily required to have some understanding of how servers and ports work, we ran into some issues that required some background knowledge. Furthermore, completing this project took understanding of many different python libraries that we had not seen before. All of the code that was required in decrypting and encrypting made use of various libraries such as Crypto.Cipher and Crypto.PublicKey. These were interesting to learn about and furthered my knowledge in these topics.