Project 2 Report

Qiuyang Fu

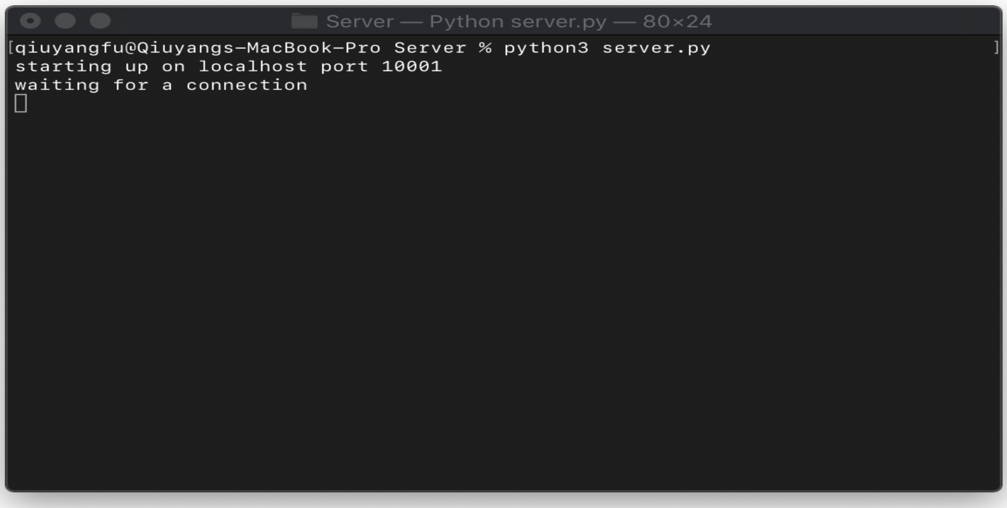
Kaiwen Chen

This project contains 3 python files. For each python file, we completed many functions to make the file runnable. After completing those three python files, we run it to generate some files we need. For add_user.py, our purpose is to add users and hashed their password to the passfile.txt. For server.py, it is to allow the client to continue. For client.py, we are going to enter the username and password and then connect to the server at the beginning of this project. We hash the password to continue. Hashed_password is used to encode the user-provided password in a way that is safe to store on disk or into a database or file. It first generates some random salt that should be added to the password. The salt is encoded as plain ASCII, and the hexadecimal representation of a hash only contains 0-9 and A-F. While the password is encoded as utf-8, it could contain any character. We learn that sha-256 is the safest way to hash in the lecture. Sha-256 hash of some random bytes read from os.urandom and then extracts a string representation of the hashed salt as a set of hexadecimal numbers. And also, the instructor mentions that MD5 is very dangerous, and sha-1 is rarely used. So we choose the sha-256, which is the best way to make our password storage algorithm secure.

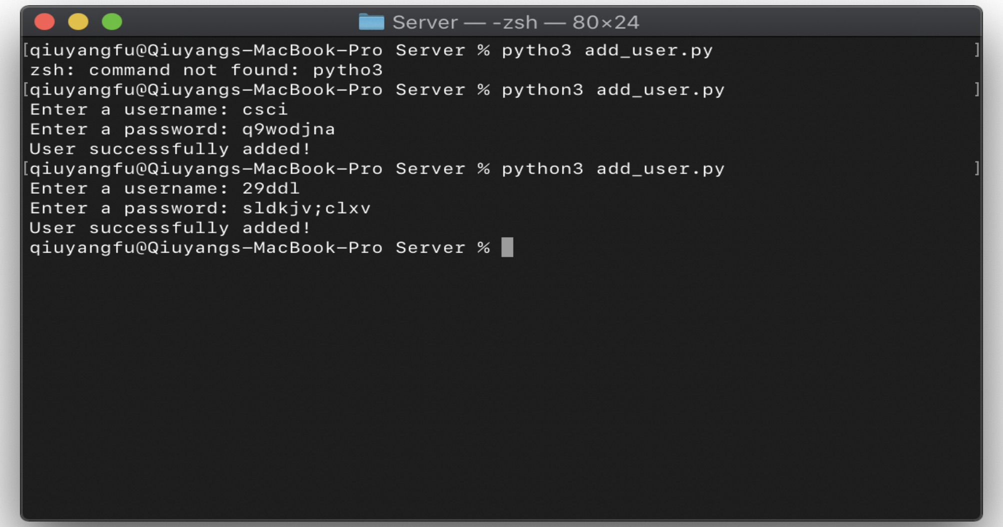
To use public/private keys to authenticate. We used command ssh-keygen-trsa to generate public/private keys. And then create a public/private key pair on the client. This command will deposit the key pair into the ssh directory. And then encrypt and send it to the server. ssh-keygen can create keys for use by ssh protocol. If invoked without any arguments, ssh-keygen will generate an RSA key. We use Crypto as a library to encrypt messages. In the Client folder, there is a .pug file to store keys. In the Server folder, we generate an RSA key file.Symmetric encryption uses the same key for both encryption and decryption. For algorithms, if it takes a long time for a person with a given key size to decrypt the message. We will say it is very safe. Used symmetric key algorithms to encrypt ciphers to protect the confidentiality of data while it's in transit. The mode we used is the block mode. I believe it's better because it uses a block cipher to provide information security and authenticity, and it uniquely requires an initialization vector which secures the data even more. The program is secure if I were to run it on a publicly visible network since all of the data that are in transit in the network are being encrypted and protected. There are also keys that prevent others who wish to break the encryption.

This project helps us to learn a lot about hashing passwords and client-server interaction. And also, we spend a lot of time generating a public/private key. We enjoy the research process when we meet challenge functions but lack of knowledge. And also, it is interesting to choose a library to finish this project. We need to keep studying in the future. We are the first time to complete a project about client-server interaction, which is an excellent experience for our group.

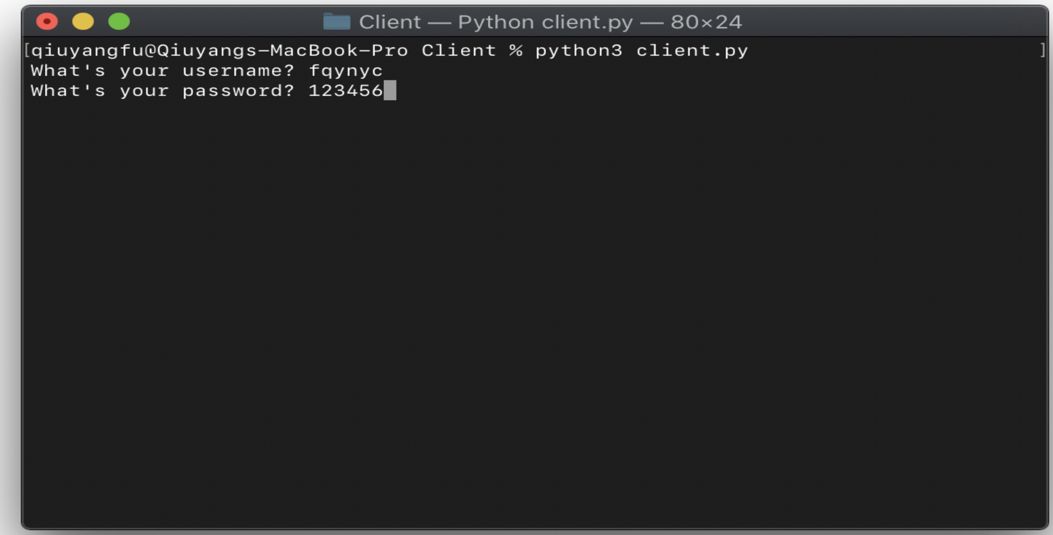
Some Screenshots:



```
qiuyangfu@Qiuyangs-MacBook-Pro Server % python3 server.py
starting up on localhost port 10001
waiting for a connection
```



```
qiuyangfu@Qiuyangs-MacBook-Pro Server % pytho3 add_user.py
zsh: command not found: pytho3
qiuyangfu@Qiuyangs-MacBook-Pro Server % python3 add_user.py
Enter a username: csci
Enter a password: q9wodjna
User successfully added!
qiuyangfu@Qiuyangs-MacBook-Pro Server % python3 add_user.py
Enter a username: 29ddl
Enter a password: sldkjv;clxv
User successfully added!
qiuyangfu@Qiuyangs-MacBook-Pro Server %
```



```
[qiuyangfu@Qiuyangs-MacBook-Pro Client % python3 client.py                    ]
 What's your username? fqynyc
 What's your password? 123456
```

Citation:

https://docs.python.org/3/library/hashlib.html

https://www.vitoshacademy.com/hashing-passwords-in-python/