Zach Schmidt

CSCI 4830-801

Final Submission

GUIDE FOR RUNNING:

System should be run with Python3

Example command:  
python .\secureShell.py --home=C:\FRESH\_DIR --user=admin --password=password1

At this moment there is something wrong with the logging configuration as well as adding and removing authorized users. The concepts are in place but the log file is M.I.A. and adding and removing authorized users has some \n related issues that cause the text to be on incorrect line.

I spent a lot of time trying to debug these issues and still havent succeeded. I decided to put the concept into place with flaws in implementation.

All commands other than authorize and deauthorize work as expected.

Both of the broken commands are very close

System security questions

a)

The most obvious flaw to me in the secure system is the command line implementation. The write up states that the entire command line will be shown in the output to ps. Other users would then be able to see the input username at password at program run time. Another weakness of the system is the lack of an integrity check on the users file. An intruder an insert themselves into the users file to give themselves access to the system. While they would be unable to view files that are encrypted, they would be able to treat the system as a normal user would. Additionally as part of my implementation, there is no constraint on password choice. With this in mind, using a weak password and no time restrictions, an intruder could easily gain access to the full system functionality.

b)

My first improvement is to run the program without arguments and only accept input from within the program, avoiding the ps problem described above. Next I would set password requirements for length and a check for single dictionary words. I also think it would be a neat feature to give the user feedback about the complexity of their password before it's accepted (example: on your current system your password would likely be broken in under x minutes.) With the time just mentioned in mind, implementing an increasing cool down period after each incorrect attempt could boost the challenge of breaking into the system. The last thing I would do is to encrypt all logs and directories used exclusively by the system. Combined with an integrity check, this would keep individuals with malicious intent from accessing the system through manipulation of users and file databases. Since the system is very simple at the moment, I would also go with the maximum encryption standards available as well as increasing the number of hash iterations for the passwords.