Read Me

1.) Members:

Vikram Krishna, NetId: Vk235, RUID: 152007297

2.) Installation:

I ran this program on an Python IDE, Jetbrains I am not 100% sure how to run on the Ilab machines, but my guess is to run the server and client, in addition the PROGRAMS WERE TESTED ON PYTHON 2.7(which I am fairly sure the Ilabs version is up to)

$python Server.py

$python Client.py

3.) Running the Program:

In order to run the program, a few steps have to be followed.

1. Place Server.py and Client.py in the same folder(or different if you want)
2. Place the certificate and private key in the same folder as the Server.py, and the certificate in the same folder as the Client.py
3. Run Server, then Client
4. Pass in the login in credentials, or new credentials if making a new account
5. Then using any of the 5 Commands, do what you desire
6. The commands are GET,POST,END,HELP,CLOSE,NON with the last 3 being used for debugging purposes, HELP explains what the other 5 messages do, CLOSE is for formally shutting down the socket(to prevent any exceptions being raised) and NON is do nothing

4.) Program Report:

There are many security features within the program. First of all, there is a password check with the associated login in info, after 3 incorrect tries, you are unable to take any action for a period of around 5 minutes, this is to simulate being rate limited. In addition, openssl is being run on top of the Tcp channel, hence encryption is also provided. In addition, all of the username, and passwords are hashed with Sha512, and salted. No need to worry about overflow since this is Python and not C. In addition, an attacker is unable to access the Groups or Messages textfiles, along with the password and username text files. Blank inputs are not allowed when Posting or Getting, invoking GET on an invalid group just creates an empty text file, but this group can still be POSTED to later on. I did not particularly worry about whether or not the certificate is valid, if the server certificate does not match what the client gets, an exception is throw, so the OS takes care of it for me.

I was however unable to find a good way of dealing with idle sockets, hence if an

attacker keeps opening up new accounts and connections, they will remain open, and most likely, external libraries need to be used.

The only unique decisions that are made, are to separate the password and login file into 2 different files, each also being hashed and salted. This is also for added security, but also means deleting accounts has to be done from the user side. In addition, each group is also represented by a textfile, this was done for simplicity since the main focus of the assignment is security principles. In addition, by using the “a+” flag in python for opening file, it lifts the burden of having to worry whether or not the file was created before

The rest of the program is a basic standard, multi threaded server and client programs. They communicate by speaking over a socket.