COMS 4180 Group 7

Jianpu Ma (jm4437) Kevin Liu (kll2146) Ilan Buchwald (ib2348) Ruijia Yang (ry2277)

Installation/User Guide

Installation

Creating and setting up certificates:

We have included some premade client and server keys/certificates in the auth folder, but you can also generate your own in a directory of your choosing.

The following instructions specify how to generate the client's private and public keys and certificate. To generate these for the server, replace `client` with `server`. - `openssl genrsa -out client.key 2048`: Generates the client's private key - `openssl req -new -key client.key -out client.csr`: Generates a certificate request for the client. Follow the prompts. - `openssl x509 -req -sha256 -days 365 -in client.csr -signkey client.key -out client.crt`: Client self-signs the requested certificate - `openssl x509 -pubkey -noout < client.crt > clientpubkey.pem`: Extracts the client's public key from the certificate. Necessary for PyCrypto - `rm *.csr`: Removes the now-unnecessary certificate request.

Note: The commands above do not set up a password for encrypting the RSA keys.

Reference: https://devcenter.heroku.com/articles/ssl-certificate-self

Environment Setting

If you are running the server on a Google VM and want to connect with a client that is running somewhere other than the VM that the server is running on, specific firewall rules need to be added to allow the TCP port connection. The process is:

1. Go to https://console.cloud.google.com and navigate to your project. On the hamburger menu in the top-left, choose "Networking" under the "Compute" section
2. Choose Firewall rules, click create new firewall rule
3. In the "Allowed protocols and ports" field, enter "tcp" to allow all TCP ports, or tcp:xxxx for a specific xxxx port
4. Under the "Source filter" dropdown, select "allow from any source" (or, you can instead designate an allowed IP range)
5. Give this rule a name in the "Name" field
6. Click Create

SSH into your Google Compute Engine VM. If it is a fresh VM (fresh Ubuntu 16.04 VMs will need this step), or if it doesn't have Python's pycrypto library, install it: - Install pip by running - `$ sudo apt-get install python-pip` - Then install the pycrypto package with - `$ pip install pycrypto` In particular, client.py uses the pycrypto package, so whatever machine you run the client on, it needs to have pycrypto installed.

Run

Quick Run: Run one of the Makefile test command pairs (`make s1` and `make c1`, `make s2` and `make c2`, etc.)

Alternatively: `python client.py <server's IP or hostname> <server port> <client certificate file path> <client private key file path> <server certificate file path> <client public key file path`

```
python server.py <server port> <server certificate file path> <server private key file path> <client certificate file path>
```

where - `<server port>` is a port number in the inclusive range [1024, 65535] to listen on - `<server certificate file path>` is the server's certificate file (for example, auth/server.crt) - `<server private key file path>` is the server's RSA private key file (for example, auth/server.key) - `<client certificate file path>` is the client's certificate file (for example, auth/client.crt) - `<client private key file path>` is the client's RSA private key file (for example, auth/client.key) - `<client public key file path>` is the client's RSA public key file (for example, auth/clientpubkey.key)

From here, the program follows the assignment specs.

Note: you should start the server before starting the client. If you start the client first and the server isn't running, the client gives an error message and exits gracefully.

Note: All files sent by the client will be put into the simulated directory structure under client_files, which is located in the same directory that the server is run in. Clients can specify absolute or relative paths for the files they 'put' or 'get'. Both will be converted into the absolute path. Then, the file the client 'put's or 'get's and its absolute path will be sent to the server, where the file will be located inside the simulated directory structure under client_files/[absolute path provided by client]. In summary, when a client 'put's or 'get's files, it does so within a simulated directory structuring mirroring the directory structure of its own system.

Note: To kill either the server or client, you can hit CTRL+C on the terminal. If the client is currently connected to the server and you exit the client (either by entering the "stop" command or hitting CTRL+C), the client exits and closes the socket on its end. This causes the server to immediately exit, as it detects that the socket was closed on the remote (client) side. If the client is currently connected to the server and you exit the server (by hitting CTRL+C), the client side still waits for user input. As soon as the user enters a valid get/put command (if the user enters "stop", the client exits, of course), the client detects that the socket was closed on the remote (server) side, and the client exits.