Name: Rachel Wilbanks

Assignment: CS465 Project1 Report

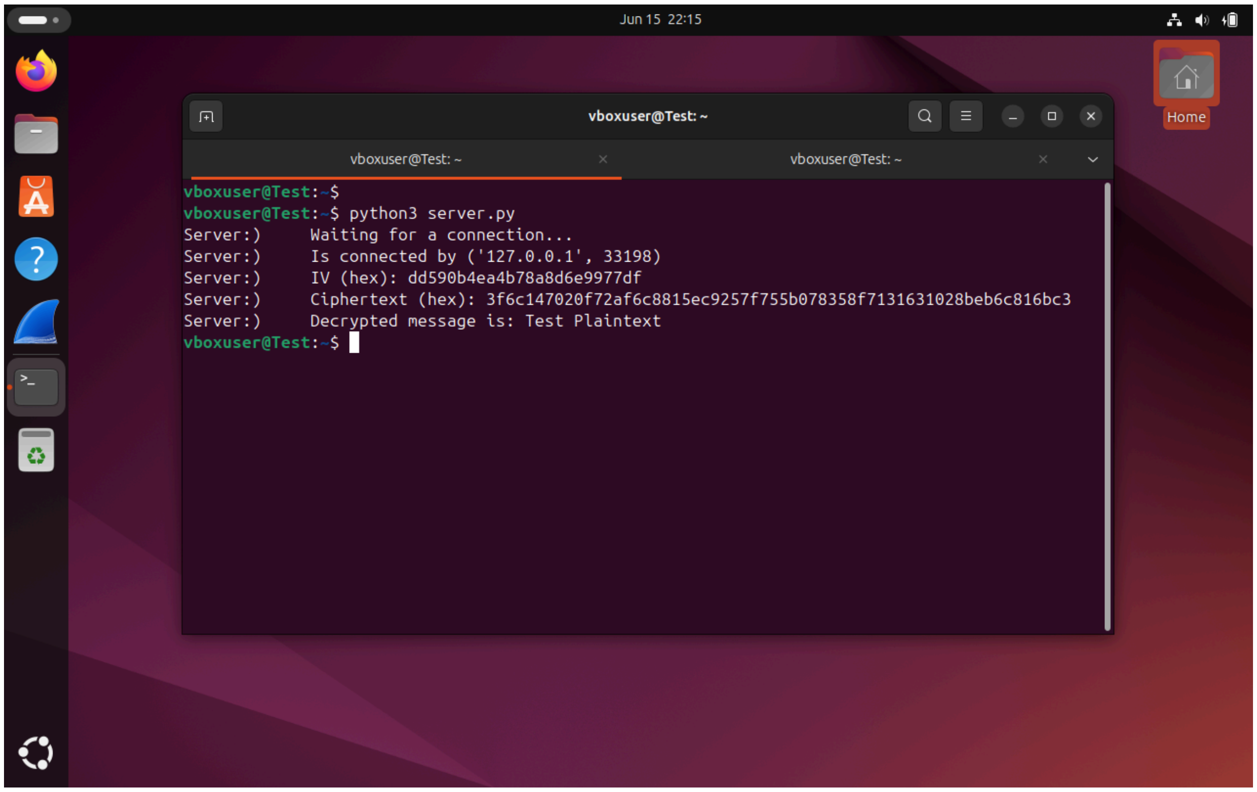
Project1: TCP Client/Server System

For this project, the goal was to create a client and server pair, over TCP, that includes endpoints that perform an exchange of keys, and encrypts the messages after the handshake. There were 3 different options given to perform this task. I chose Option A, which requires that both the client and server are pre-configured with the same key, and that either AES-CBC or AES-GCM are used with proper IV handling. This is accomplished by using tools found in Python libraries (cryptography, socket, os, etc.), which are necessary, at least in this project, to set up a TCP socket, refer to AESGCM, encrypt and decrypt the message, create an IV, etc. More specifically, a key is determined, plaintext is given, said plaintext is decoded and encrypted, the server receives said text, and said text is decrypted by the server (the plaintext, ciphertext, IV, decrypted message, and a confirmation message are given during the program's run).

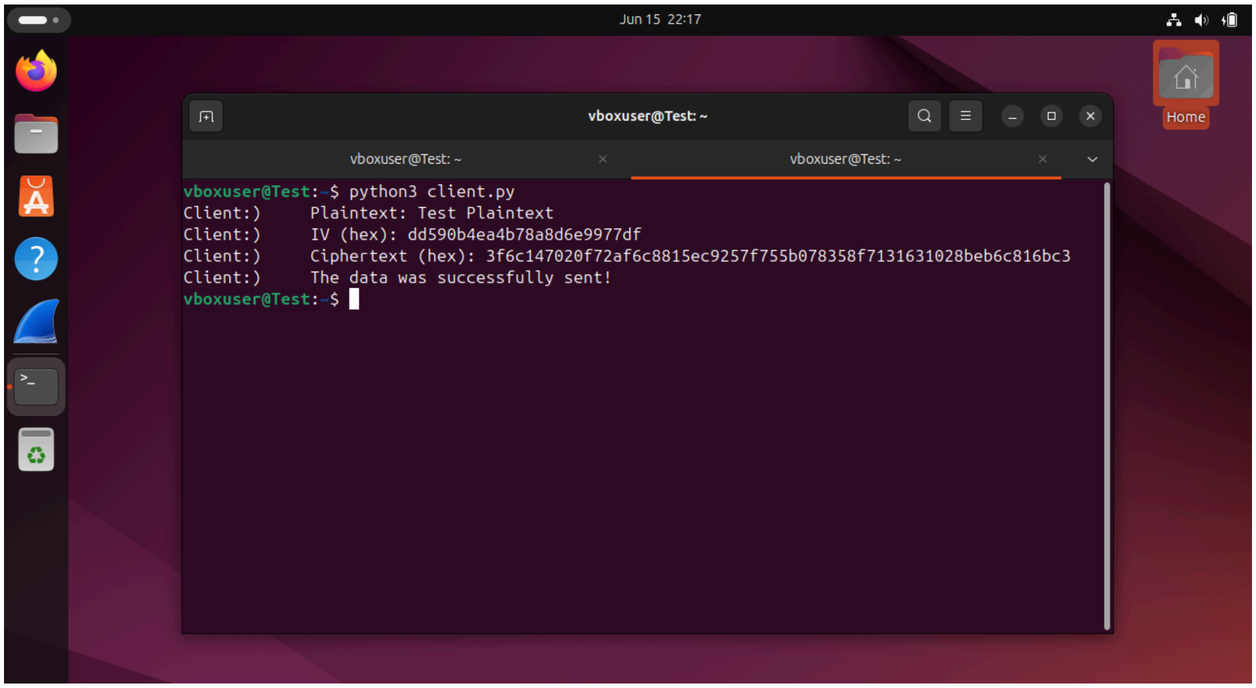
When it comes to security, AES-256 (considered to be secure) was used in this program to assist in encryption of the message. A key is shared between the client and server. AES itself is secure, but the fact that the key is coded into both programs creates a scenario where everything can be seen if the code is shared or leaked. Also, neither the client or server have any specific method of authentication, so outsiders could impersonate the client or server as long as they have access to the key.

The main challenge I had with this project came from remembering exactly how to use Python's cryptography, socket, and os libraries (though I could sort of recognize said libraries, I was not able to remember much off the top of my head, which led to a lot of research, reading, and catching up on my behalf). The next challenge had to do with picking which encryption method to use for the project. I ended up picking Option A (Symmetric Encryption with Pre-Shared Key), but did experiment with Option B (Asymmetric Handshake + Symmetric Encryption) a bit before deciding to do Option A. My output (and code in general) also looked pretty messy as well, which I decided to fix after figuring out how to make it more legible to an outside viewer (I added more comments as well, but that was more for me than it was for others viewing the code, as I like to comment things in unnecessary detail just in case I need to go back and reference or use this code long after writing it). Overall, I learned how to use multiple Python libraries, how to properly use socket programming, how encryption and decryption can be accomplished using code, how IV is used in the client-server process, and more.

Program Screenshots



```
vboxuser@Test:~$
vboxuser@Test:~$ python3 server.py
Server:)     Waiting for a connection...
Server:)     Is connected by ('127.0.0.1', 33198)
Server:)     IV (hex): dd590b4ea4b78a8d6e9977df
Server:)     Ciphertext (hex): 3f6c147020f72af6c8815ec9257f755b078358f7131631028beb6c816bc3
Server:)     Decrypted message is: Test Plaintext
vboxuser@Test:~$
```



```
vboxuser@Test:~$ python3 client.py
Client:)     Plaintext: Test Plaintext
Client:)     IV (hex): dd590b4ea4b78a8d6e9977df
Client:)     Ciphertext (hex): 3f6c147020f72af6c8815ec9257f755b078358f7131631028beb6c816bc3
Client:)     The data was successfully sent!
vboxuser@Test:~$
```