CNT4713 Cryptography Project

Overview

In this project you will create a network application using Python which will allow clients to send encrypted messages between each other. There are two parts to this project:

1. The Server
    a. On start, the server will create a public-private key pair and store it in memory.
    b. The server will use TCP to create a listening socket on a usable port. You may use 8080.
    c. The server will listen for the following **commands** (in bold) from the client and:
        i. **connect** – The server will establish the connection with the client and respond with a different port number for the client to accept data connections. The server creates a new socket for this data connection.
        ii. **tunnel** – The server accepts the client's public key. The server responds on the data socket with its own public key.
        iii. **post** – The server accepts an encrypted message and decrypts it using the correct key. The server computes the SHA256 hash of the message, encrypts it using the correct key, and responds to the client with the encrypted hash (the cipher text). The server responds on the data socket.
2. The Client
    a. On start, the server will create a public-private key pair and store it in memory.
    b. The client will create a socket to connect to the server using the **connect** command. On server response, the client will create a new socket for the data connections.
    c. The client will send the server its public key using the **tunnel** command. The client will store the response from the server for future use.
    d. The client will create a message, encrypt it using the server's public key, and send it to the server using the **post** command.
    e. On response from the server, the client will compute a SHA256 hash of its original message and compare it to the response from the server. If they are equal, print "**Secure**". If they are not equal, print "**Compromised**".

Output

Use the following output guideline using the message “Hello” through the stages of the appications:

1. The Server

```
Starting server…

Creating RSA keypair

RSA keypair created

Creating server socket

Awaiting connections…

Connection requested. Creating data socket

Tunnel requested. Sending public key

Post requested.

Received encrypted message: <Encrypted output>

Decrypted message: Hello

Computing hash

Responding with hash: <Hash of Hello>
```

2. The Client

```
Starting client…

Creating RSA keypair

RSA keypair created

Creating client socket

Connecting to server

Creating data socket

Requesting tunnel

Server public key received

Tunnel established

Encrypting message: Hello
```

```
Sending encrypted message: <Encrypted output>

Received hash

Computing hash

Secure
```

Submission

1. In a text file, answer the following question(s):
    a. How could the “secure” server and client framework above be compromised?
    b. How could you improve the framework?
2. Submit a copy of your final code to Canvas. (Source code only. No zips. No IDE project files)
3. A video recording (screen recording) of the execution of your team's project. The recording must show your application being executed for each graded part of the project. Narrate the video stating what is being shown.

Grading

Item	**Percentage**
connect command	20
tunnel command	20
post command	20
Displays **Secure**/**Compromised** correctly	20
Answers to questions	15
Video	5