**PBPROXY**

* **Introduction**

This application’s main goal is protecting the Server by encrypting/decrypting the data required by the server. It consists of two parts: PBProxy Server and PRProxy Client. It is run as follows:  
Client : pbproxy -k mykey 127.0.0.1 22  
Server: pbproxy -k mykey -l 2222 127.0.0.1 22  
When run without using the “-l” flag, this application acts as PBProxy Client and encrypts the data by AES CTR encryption using a key mentioned in the file “mykey” (mentioned in the command followed by -k flag) and by generating a random Initialization Vector. This message is then decrypted by the PBProxy, listening on the port specified by “-l” flag, using the key in the file “mykey”, and then forwarded to the Server. The PBProxy server also encrypts the messages from the actual server and sends them to the client, which decrypts them and displays on the screen. The parameters -l, -k and the host/port can be used in any combination.  
  
Example usage of this application:  
Server: pbproxy -k mykey -l 2222 localhost 22  
Client: ssh -o "ProxyCommand <fullPathToProxy>/pbproxy -k <full or relative path to this file>/mykey 127.0.0.1 2222" localhost

* **Design Details:**  
  The server application uses two threads:
* The thread which reads from PBProxy Client, decrypts the message and forwards to Server
* The thread which reads from Server and then forwards the message to the Client

These threads keep listening for incoming data, read it as it arrives and then transfers it to the destination.

I am using OpenSSl libraries for encryption/decryption and arpa/inet.h library for socket programming, and pthread.h for threading.

The following are the primary functions used in the code:

* encryptionSetup – Creates an IV, send it to PBProxy Server and initiliazes the structure used for encryption
* decryptionSetup – Receives an IV from Client, initiliazes the structure used for decryption
* createServerSocket – Creates a server socket which listens to the port specified in the argument
* createClientSocket – Creates a client socket which sends data to the port and host specified as argument
* **Technical Details:**

The following are technical specifications of the project:

* Test environment: Ubuntu 16.04.3 (Linux 4.10.0-38-generic x86\_64)
* Compiler and its version: gcc 5.4.0
* **References:**  
  <http://www.gurutechnologies.net/blog/aes-ctr-encryption-in-c/>

<https://vcansimplify.wordpress.com/2013/03/14/c-socket-tutorial-echo-server/>

<http://www.geeksforgeeks.org/multithreading-c-2/>

<http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html>