CSE508: Network Security, Spring 2021
Homework 4: Plugboard Proxy

IMPLEMENTATION

There are two modes of PBProxy: (Reverse Proxy Mode and Client Mode).

- The command arguments conform to the specifications of the question. If any of the required inputs are not given, the program prompts an error message and asks the user for its input.

- If listen port is provided with -l flag, it starts in Reverse Proxy mode, else in client mode.

CLIENT MODE :

- This is fairly straightforward. The program dials a connection with the destination port and host (proxy server Ip and port in our case).
- After establishing connections, it reads from stdin, encrypts the message and writes to destination connection.
- It also spawns a go routine which continually reads from the destination, decrypts the data and writes to stdout.

REVERSE PROXY MODE :

- The program scans the devices and gets the IP address of the first device and starts listening on the host IP and the listen port.
- It then dials a connection with the destination host and port (localhost and sshd port in our case).
- It then reads messages from source connection (from client), decrypts it and writes it to the destination.
- In parallel go routine, it reads from the destination, encrypts it and sends back to the source connection (client connection).
- The reads and writes happen till EOF is encountered, after which the loop breaks.

- **With the handling of multiple listeners, I am able to run multiple SSH connections via the proxy server, i.e if one ssh connection is running, from another terminal, I can spawn another connection. The reverse proxy keeps on listening for connections in infinite loop and once accepted, passes the connections to the go routines.**

- The data is encrypted/decrypted using AES-256 in GCM mode (bi-directional communication). Hence client takes input from stdin in plaintext traffic , encrypts it and sends it to the proxy server. The proxy server decrypts it and sends it to the destination server (**sshd in our case**).

- So overall, the proxy server also reads from the destination server sockets (the response from SSH), encrypts it and sends it back to the client. The client then decrypts the message received from the proxy server. Hence, the encryption decryption is done both ways.

- To make the channel secure, I am generating the nonce and salt every time from a random generator and then sending it with the message. At the receiving end, the message is parsed, nonce, salt and encrypted data is separated and the data is then decrypted using the same nonce and salt. Hence nonce and salt are always generated afresh.

SAMPLE RUN:

---

1) On destination server run :
go run pbproxy.go -l 2222 -p pwdFile localhost 22

2) On client system, run :
ssh -o "ProxyCommand go run pbproxy.go -p pwdFile <PROXY SERVER IP> 2222" username@localhost

This also allows MULTIPLE CONCURRENT SSH connections via the proxy server. This has been tested. Please find the attached screenshot.

CAUTION:

I have tested the program to find the host system's IP address by scanning all devices.
Incase the proxy server is not able to start, please find **getLocalIP()** in line 51 and replace it with IP address of the proxy server host.

SUBMISSIONS:

---

*- pbproxy.go
*- pwdFile
*- proxy-multiplessh.png
*- hw4-report.txt

References :

https://cryptobook.nakov.com/symmetric-key-ciphers/cipher-block-modes
https://pkg.go.dev/net
https://pkg.go.dev/crypto/cipher
https://pkg.go.dev/golang.org/x/crypto/pbkdf2
https://medium.com/@yanzay/implementing-simple-netcat-using-go-bbab37507635