Implementation Steps:

I have a file pbproxy.go, which follows the following specification:

go run pbproxy.go [-l listenport] -p pwdfile destination port

1. pwdfile, destination and port are all mandatory fields.
2. If listenport is provided, the program starts in Reverse Proxy Mode. If not provided, it starts in Client Mode.

Client Mode :

1. Client connects to the provided destHost and destPort. (Socket is created using net.Dial()) [function name -> startClient]
2. It runs with two goroutines -
   1. It continuously reads from stdin and the input is then encrypted and sent to the server via socket. [function name -> readFromStdin]
   2. It continuously polls the server and reads and decrypts and writes to stdout.

[function name -> writeToStdout]

Reverse Proxy Mode:

1. Server binds to the listenPort to look for incoming connections from the client. It also connects to the target service using destHost and destPort. [function name -> startServer]
2. It runs with two goroutines -
   1. It continuously listens for new connections from listenPort, decrypts the data and forwards it to the target service [function name -> processClient]
   2. It continuously polls the target service , encrypts the data and forwards it to the client. [function name -> processReverseServer]

Note:

1. I have used 1024 as the default buffer size, and if the size is >1024, before encrypting data I am splitting it into additional blocks and sending it with length. While decrypting, I add all the blocks together and decrypt. [function names -> recvInBlocks and sendInBlocks]
2. Nonce size (randomly generated) -> 12 bytes
3. Salt size (randomly generated) -> 8 bytes
4. Key is generated using pbkdf2, and encryption/decryption method used is AES-256 in GCM mode [function names -> getCipherText and getPlainText]

Examples:

**Case 1:**

Using echo server with ncat to test server/client locally :

Client command -> go pbproxy.go -p pass.txt localhost 2222

Server command -> go pbproxy.go -l 2222 -p pass.txt localhost 2234

Ncat command -> ncat -l 2234 -k -c ‘xargs -n1 echo’

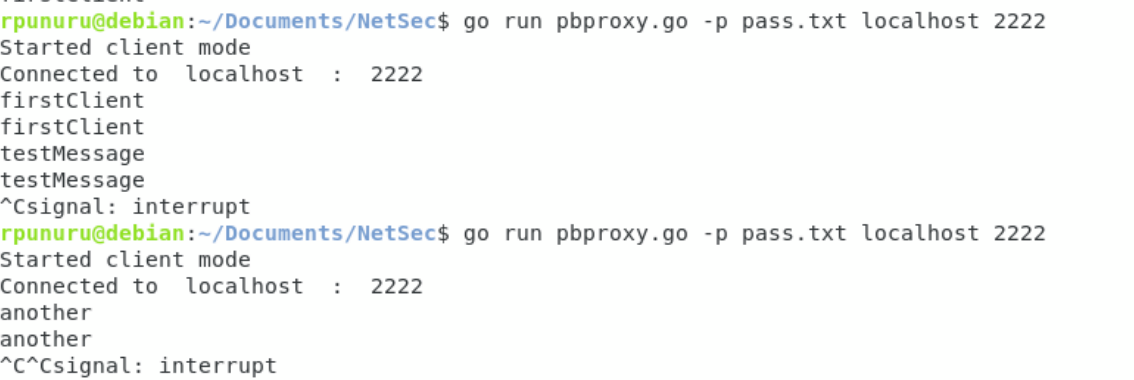
Tcpdump commands to check data flow:

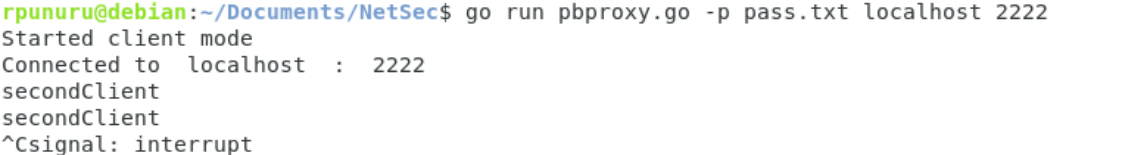
At server -> sudo tcpdump -nX tcp -i lo port 2222

At target service -> sudo tcpdump -nX tcp -i lo port 2234

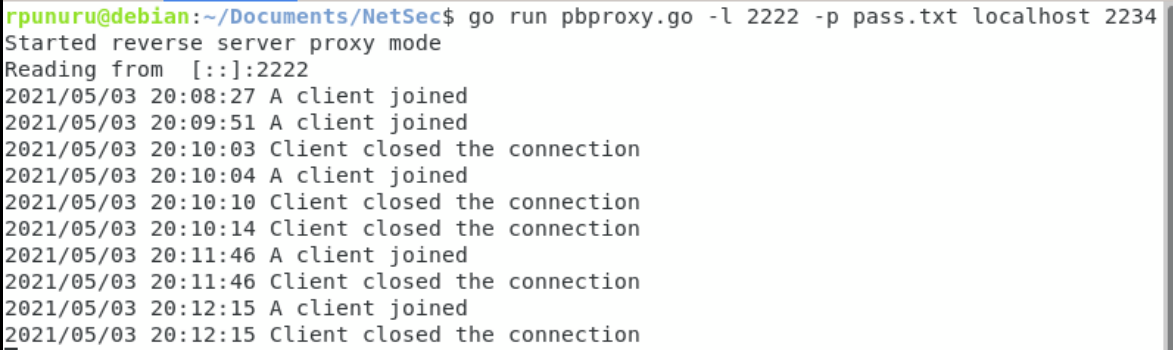


After I run the echo server, I ran the client as follows and the stdin input was echoed back.

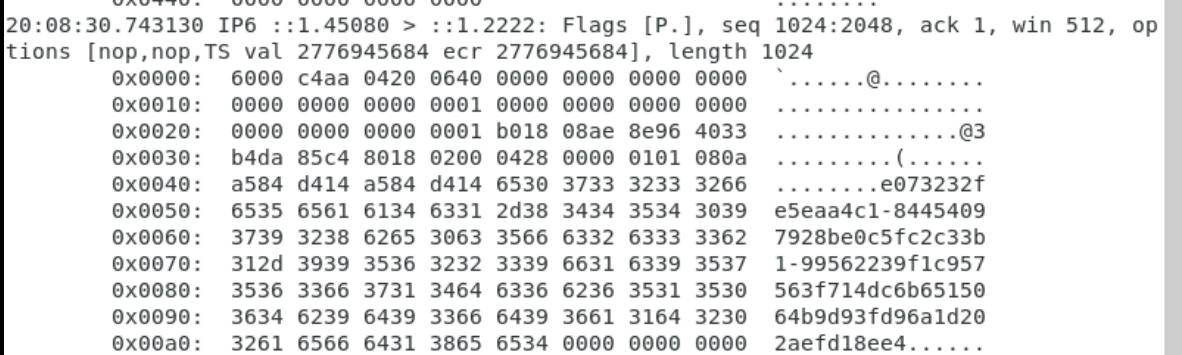




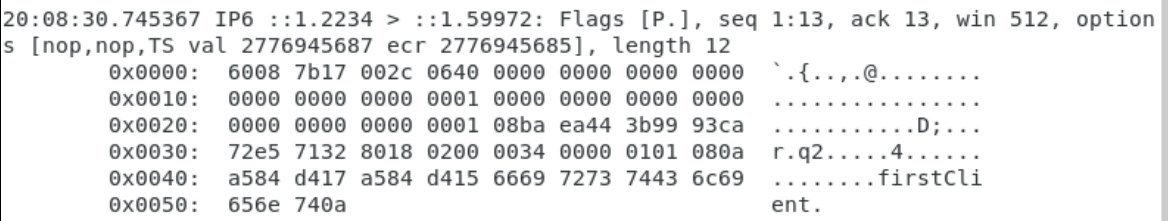
The server connects to multiple clients simultaneously.



Data in proxy server (port 2222) is encrypted

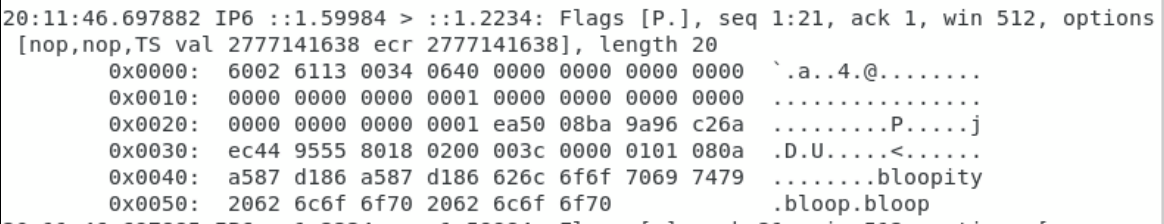


Data in target service(port 2234) is decrypted



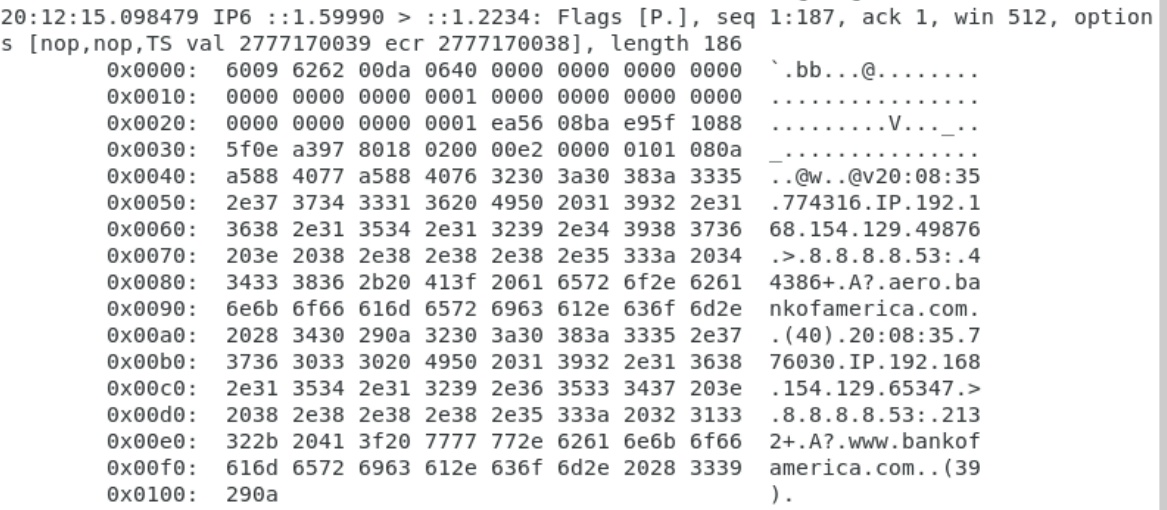
When I send a message directly to the proxy server, it is forwarded to the target server.





When I send a binary file to the proxy server, it is forwarded to the target server.



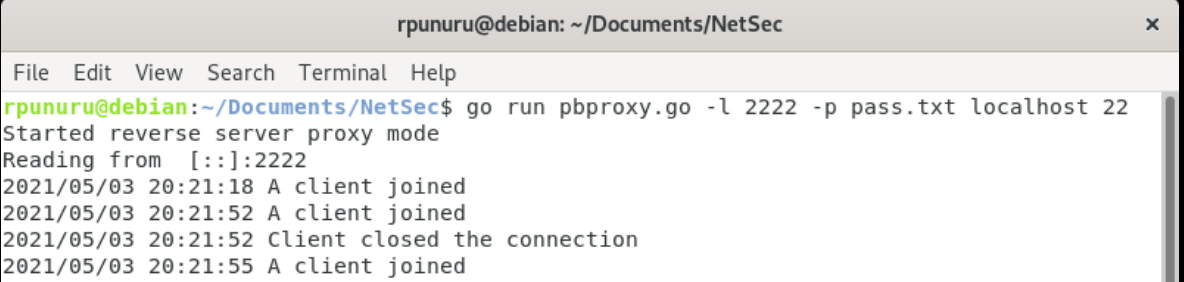


**Case 2 :**

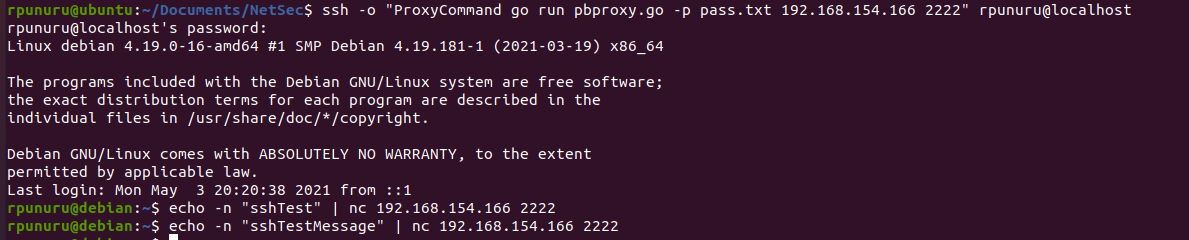
Using ssh:

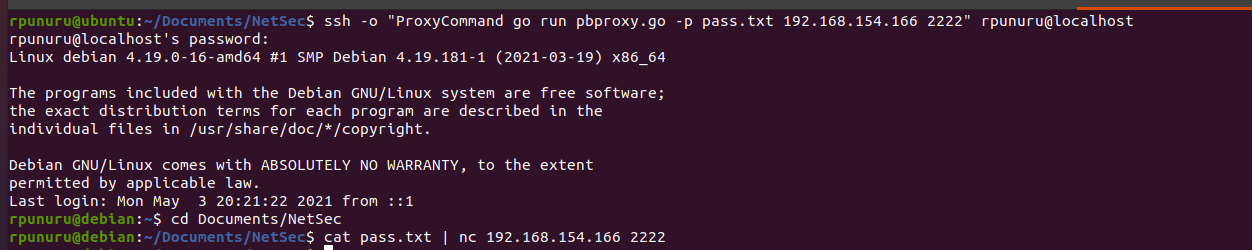
Client command -> ssh -o “ProxyCommand go pbproxy.go -p pass.txt 192.168.154.166 2222” localhost

Server command -> go pbproxy.go -l 2222 -p pass.txt localhost 22



Connected multiple ssh sessions at the same time





Data in target service(port 22)

