Networks and Systems Security
Assignment 3

Assumptions:

1. The maximum size of the encrypted/decrypted text is 1024-48=976
2. The key is known and is hardcoded

How to Run:

- Keep both the client and server in the same folder and run **make** or **make all**
- Create a text file (.txt) and in it, type out the plaintext you want to send
- Create two terminal sessions
- In one, run **./server**
- In the other, run **./client <filename>**. Replace the <filename> with the name/path of the file with the input plaintext
  A file named **output.txt** will be created
- Run **cat output.txt** to check its contents. If verification fails, its contents will read **Incorrect**. Otherwise, it will contain the same plaintext as the input file
- Run **make clean** to remove the executable and output files

The Client Code:

The code initially creates a 2-way pipe and then forks the main program, hence creating two copies of the program:

1. The original program defines a 32 byte key (hardcoded) and a 16 byte IV (by reading 16 bytes from /dev/urandom). The plaintext is then read from the input file. The plaintext is then encrypted using the defined/generated key and IV. We then create an EVP_PKEY type from our key and use it to create the HMAC. We then put the IV (16 bytes), HMAC (32 bytes) and the encoded information into the same buffer and write it onto the pipe

2. The copied program makes the other end of the pipe equivalent to STDIN, so any data written on the original side effectively goes into STDIN. We then run execl to replace the current program with the netcat binary which attempts to connect to the server

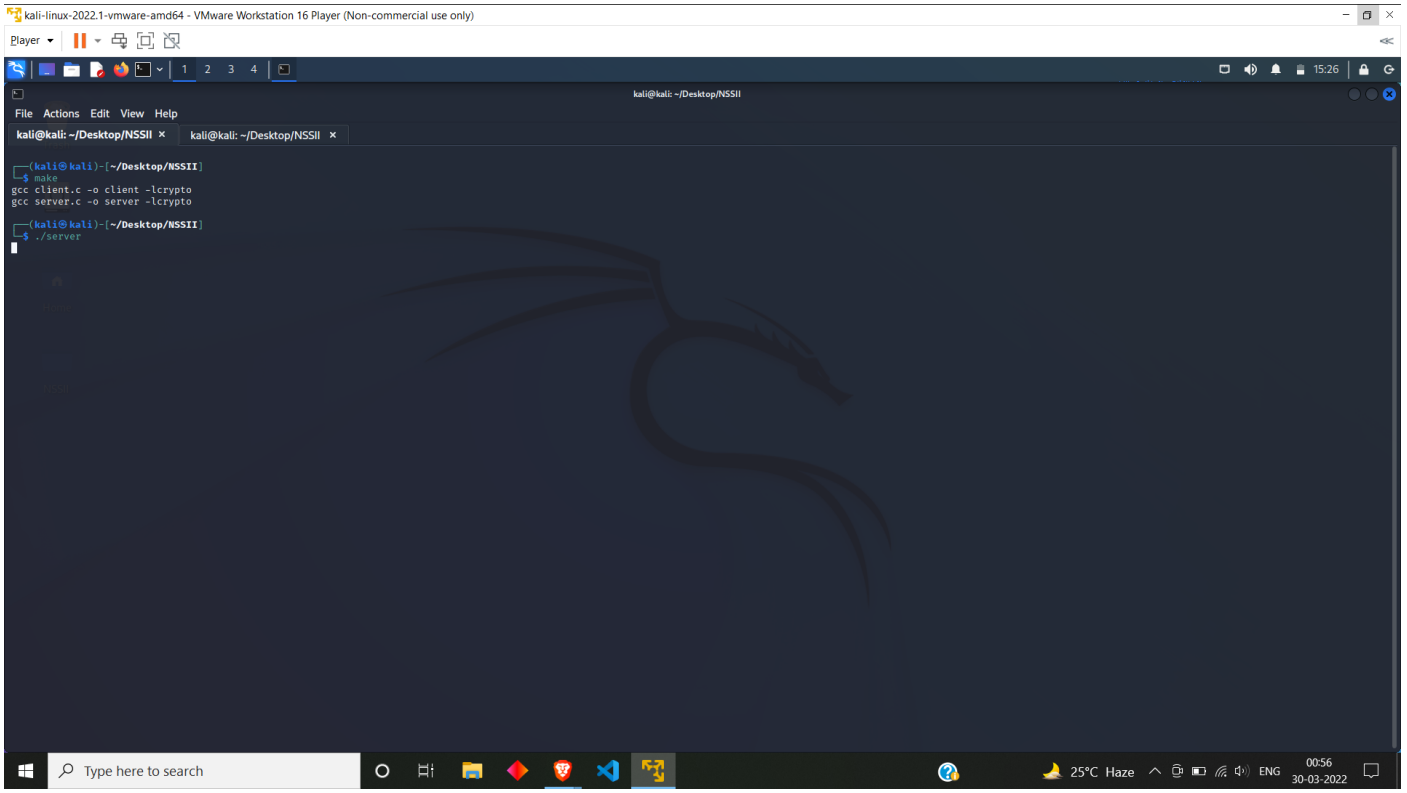
The result is that the buffer written by the original program goes into STDIN and is sent to the server via netcat

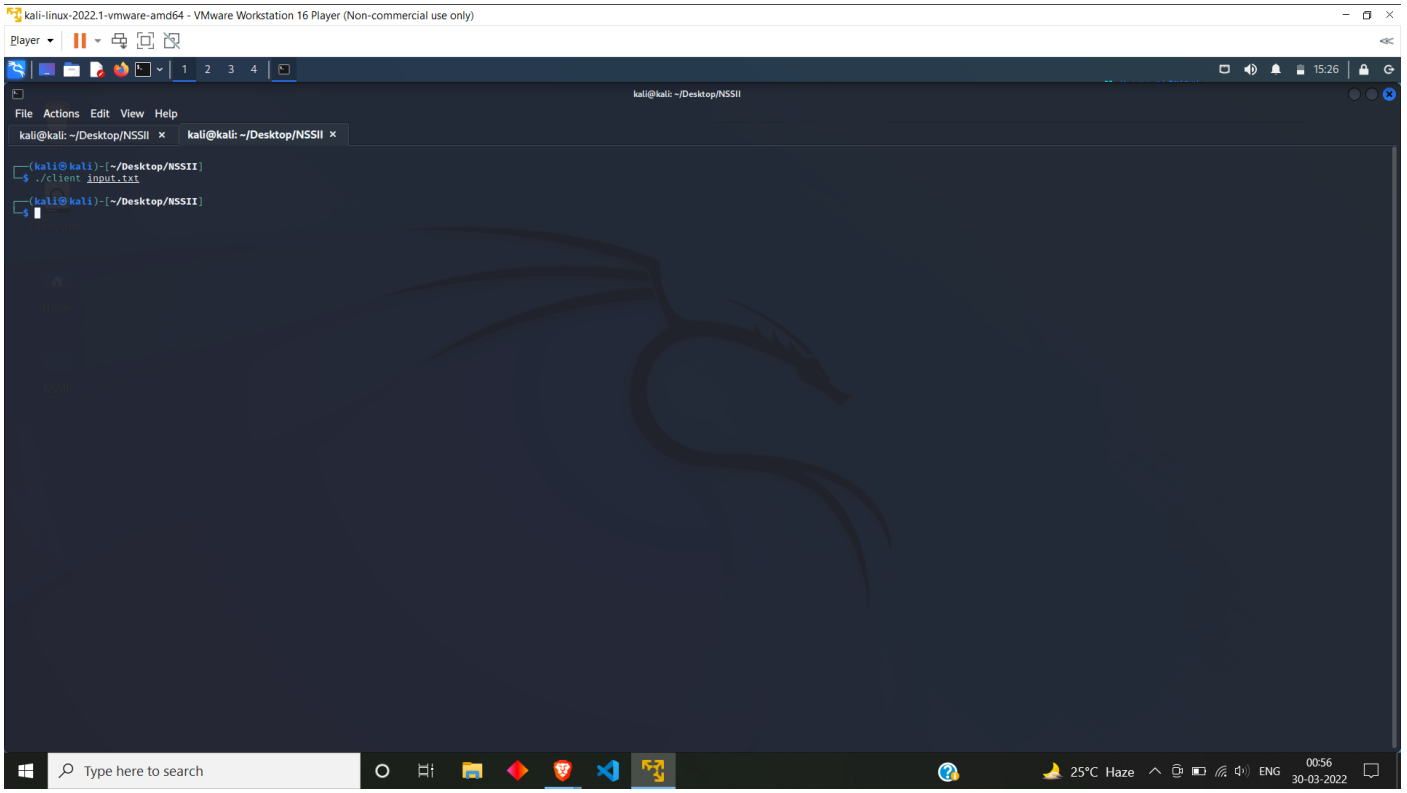
The Server Code:

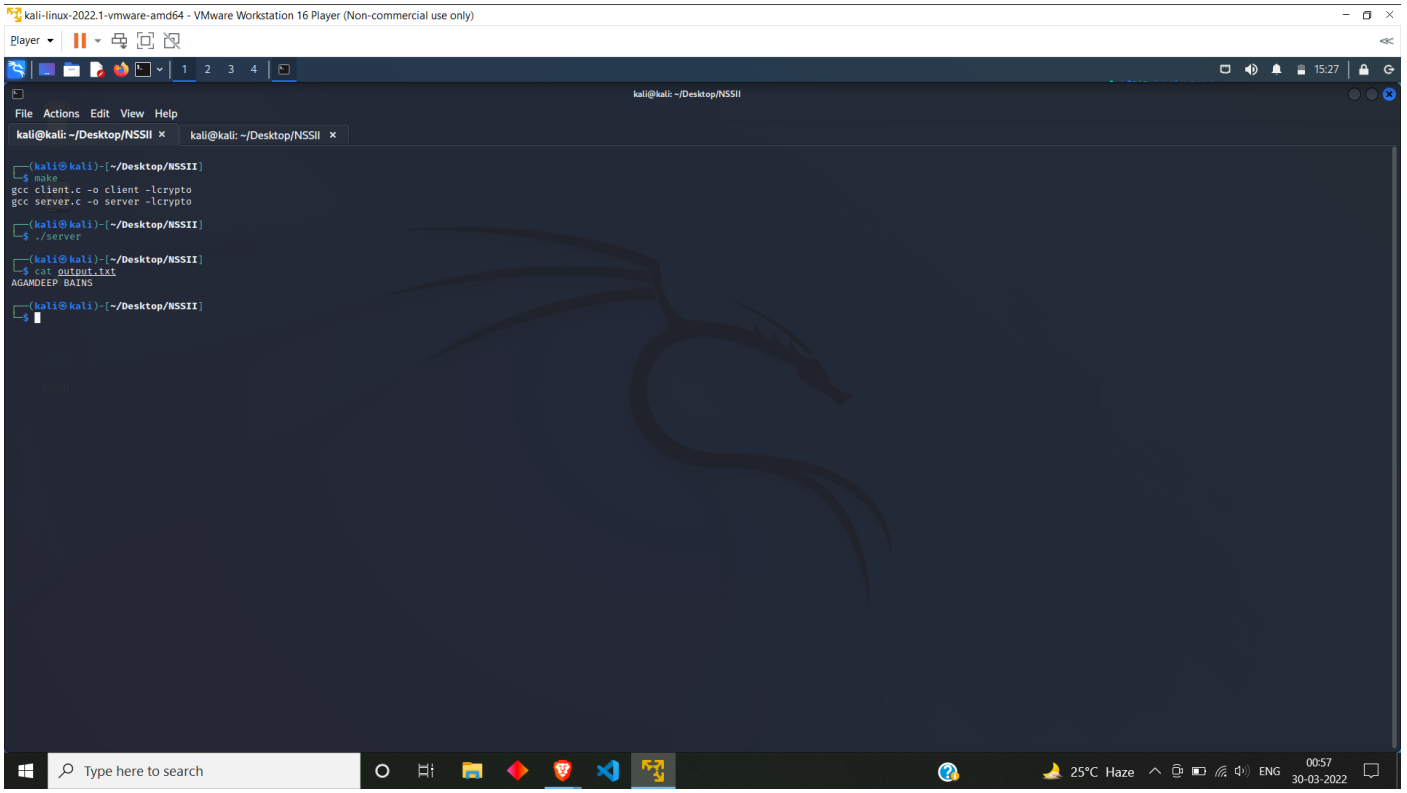
The code initially creates a 2-way pipe and then forks the main program, hence creating two copies of the program:

1. The original program defines a 32 byte key (hardcoded) and then attempts to read data from the pipe. Once the data is read, its first 16 bytes are taken to be the IV (by our convention) and the next 32 bytes are the HMAC bytes (by our convention). The rest of the read data is the ciphertext. We then create an EVP_PKEY type from our key and use it to verify the sent HMAC. If the HMAC cannot be verified, we write **Incorrect** onto the **output.txt** file. If the HMAC is successfully verified, we decode the ciphertext using the key and IV and write it into the same file
2. The copied program makes the other end of the pipe equivalent to STDOUT, so any data written onto STDOUT comes onto the original side of the pipe. We then run execl to replace the current program with the netcat binary which listens to a port

The result is that when the client connects to the netcat and send the data, it is written onto the pipe


kali-linux-2022.1-vmware-amd64 - VMware Workstation 16 Player (Non-commercial use only)
Player
kali@kali: ~/Desktop/NSSII
File Actions Edit View Help
kali@kali: ~/Desktop/NSSII
kali@kali: ~/Desktop/NSSII
(kali㉿kali)-[~/Desktop/NSSII]
$ make
gcc client.c -o client -lcrypto
gcc server.c -o server -lcrypto
(kali㉿kali)-[~/Desktop/NSSII]
$ ./server
Type here to search
25°C Haze
ENG
00:56
30-03-2022


kali-linux-2022.1-vmware-amd64 - VMware Workstation 16 Player (Non-commercial use only)
Player
kali@kali: ~/Desktop/NSSII
File Actions Edit View Help
kali@kali: ~/Desktop/NSSII
kali@kali: ~/Desktop/NSSII
(kali㉿kali)-[~/Desktop/NSSII]
$ ./client input.txt
(kali㉿kali)-[~/Desktop/NSSII]
$
Type here to search
25°C Haze
ENG
00:56
30-03-2022


kali-linux-2022.1-vmware-amd64 - VMware Workstation 16 Player (Non-commercial use only)
Player
kali@kali: ~/Desktop/NSSII
File Actions Edit View Help
kali@kali: ~/Desktop/NSSII
kali@kali: ~/Desktop/NSSII
┌──(kali㉿kali)-[~/Desktop/NSSII]
└─$ make
gcc client.c -o client -lcrypto
gcc server.c -o server -lcrypto
┌──(kali㉿kali)-[~/Desktop/NSSII]
└─$ ./server
┌──(kali㉿kali)-[~/Desktop/NSSII]
└─$ cat output.txt
AGAMDEEP BAINS
┌──(kali㉿kali)-[~/Desktop/NSSII]
└─$
Type here to search
25°C Haze
ENG
00:57
30-03-2022