



CY2004

Cyber Security

Assignment 03

TEST REPORT OF ALL THE IMPLEMENTATION.

Submitted by: Abdul Ahad

Roll number: 23i-2014

Date: 09-10-2024

Table of Contents

1. Client's cpp file:	3
2. Server cpp file:	4
Test Overview:	4
Test Scenarios:	4
1. Registration of New User:	4
2. Login of existing users:	7
Wireshark Analysis:	10
Chat Box:	12
Server-side chat encryption using AES-128:	12
Client-side chat Encryption using AES-128L:	13
Additional Functionalities:	14

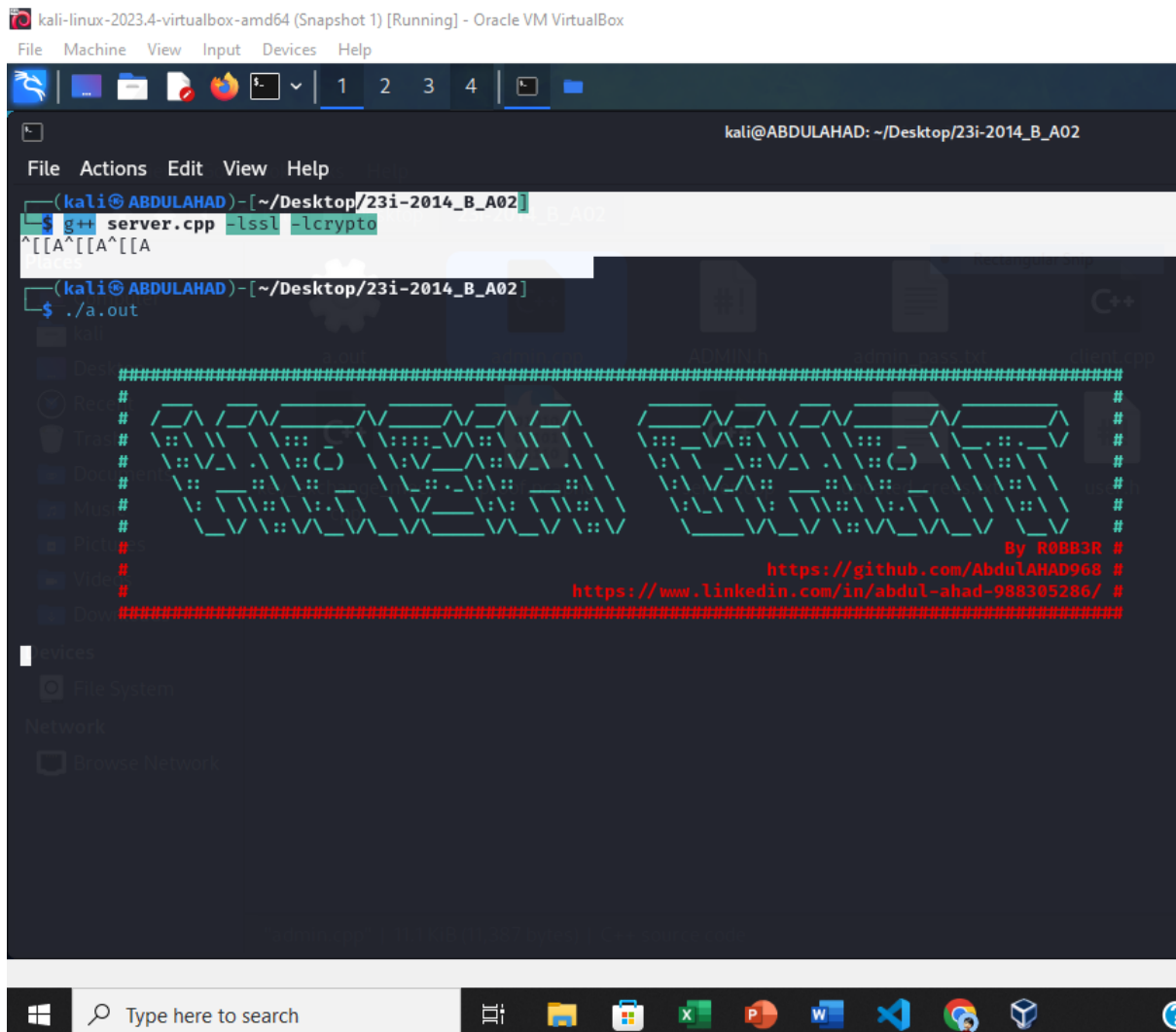
Testing all the functions.

Testing started with compiling both the .cpp files i.e.: the server.cpp and client.cpp files

1. Client's cpp file:

[illegible]

2. Server cpp file:



The screenshot shows a Kali Linux virtual machine window titled "kali-linux-2023.4-virtualbox-amd64 (Snapshot 1) [Running] - Oracle VM VirtualBox". The terminal window has a menu bar with "File", "Actions", "Edit", "View", and "Help". The prompt is "kali@ABDULAHAD: ~/Desktop/23i-2014_B_A02". The user enters the command `g++ server.cpp -lssl -lcrypto`, which compiles the program. The next command is `./a.out`, which runs the program. The output is a large ASCII art graphic of a server rack with the text "By R0BB3R" and two URLs: <https://github.com/AbdulAHAD968> and <https://www.linkedin.com/in/abdul-ahad-988305286/>. The terminal window also shows a sidebar with "Devices" and "Network" sections.

Test Overview:

I conducted a series of tests on the user registration feature to evaluate its functionality and security measures. The tests included attempts to register an existing user, checks for password strength, and validation of email structure.

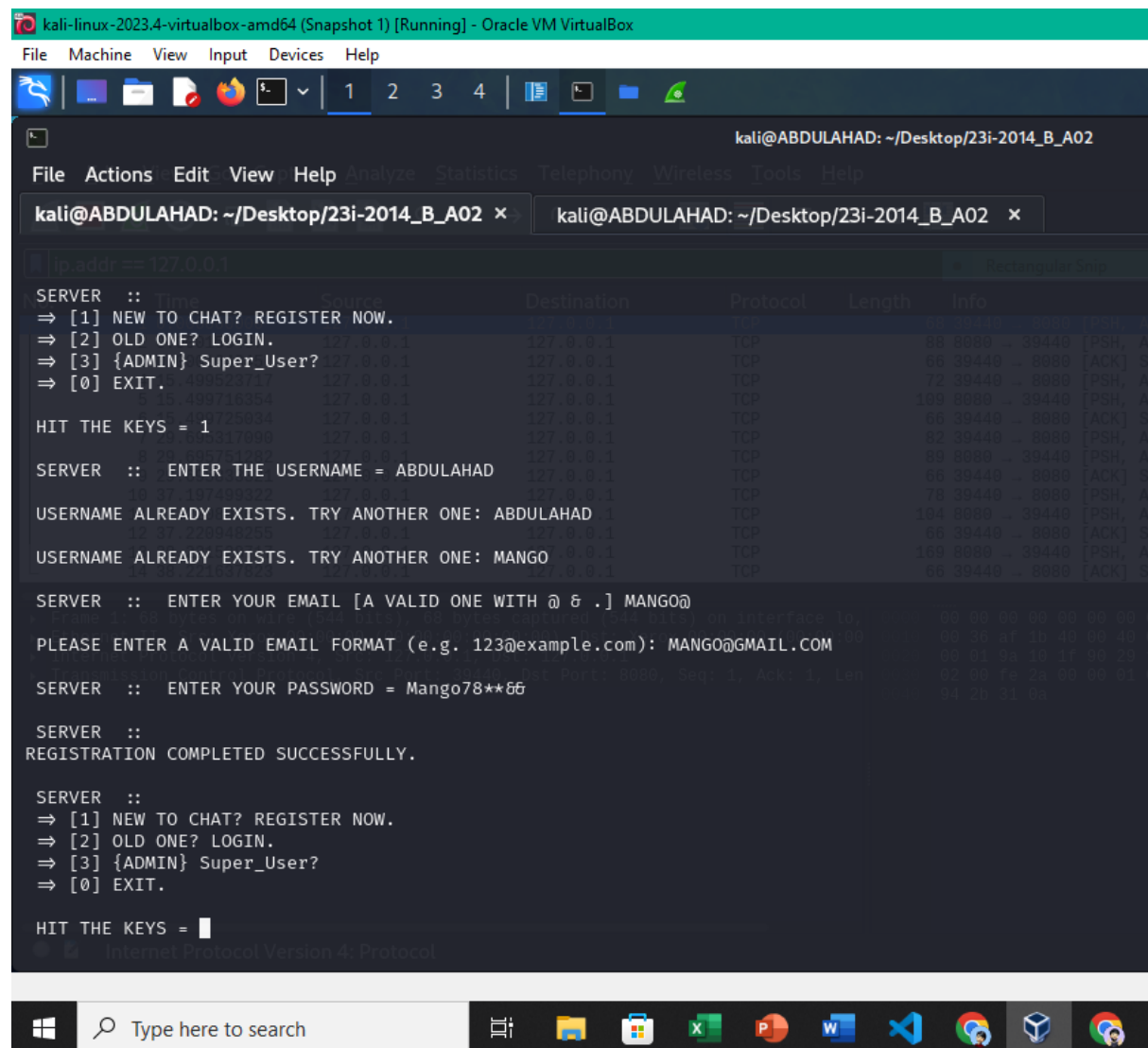
Test Scenarios:

1. Registration of New User:

Successfully registered a new user with valid credentials. Verified that the registration process is completed without errors. Duplicate User Registration:

Attempted to register multiple times using an existing user's credentials. Observed that the system correctly rejected the duplicate registration attempts, displaying an appropriate error message. I also entered email addresses with various formats (valid and invalid).

I verified that the system accurately identified valid email structures and rejected those that were incorrectly formatted.



```
kali@ABDULAHAD: ~/Desktop/23i-2014_B_A02
File Actions Edit View Help Analyze Statistics Telephony Wireless Tools Help
kali@ABDULAHAD: ~/Desktop/23i-2014_B_A02 x kali@ABDULAHAD: ~/Desktop/23i-2014_B_A02 x
ip addr == 127.0.0.1
SERVER ::
=> [1] NEW TO CHAT? REGISTER NOW.
=> [2] OLD ONE? LOGIN.
=> [3] {ADMIN} Super_User?
=> [0] EXIT.
HIT THE KEYS = 1
SERVER :: ENTER THE USERNAME = ABDULAHAD
USERNAME ALREADY EXISTS. TRY ANOTHER ONE: ABDULAHAD
USERNAME ALREADY EXISTS. TRY ANOTHER ONE: MANGO
SERVER :: ENTER YOUR EMAIL [A VALID ONE WITH @ & .] MANGO@
PLEASE ENTER A VALID EMAIL FORMAT (e.g. 123@example.com): MANGO@GMAIL.COM
SERVER :: ENTER YOUR PASSWORD = Mango78**66
SERVER ::
REGISTRATION COMPLETED SUCCESSFULLY.
SERVER ::
=> [1] NEW TO CHAT? REGISTER NOW.
=> [2] OLD ONE? LOGIN.
=> [3] {ADMIN} Super_User?
=> [0] EXIT.
HIT THE KEYS =
```

I tested various passwords to assess the system's response to weak passwords. Confirmed that the system prompted users to create stronger passwords when weak ones were entered, ensuring enhanced security.

```
HIT THE KEYS = 1

SERVER :: ENTER THE USERNAME = er

SERVER :: ENTER YOUR EMAIL [A VALID ONE WITH @ & .] er@.

SERVER :: ENTER YOUR PASSWORD = hel

SERVER :: WEAK PASSWORD! [A-Z] - [a-z] - [0-9] - [special]
SERVER :: ENTER YOUR PASSWORD =
HIT THE KEYS = hell

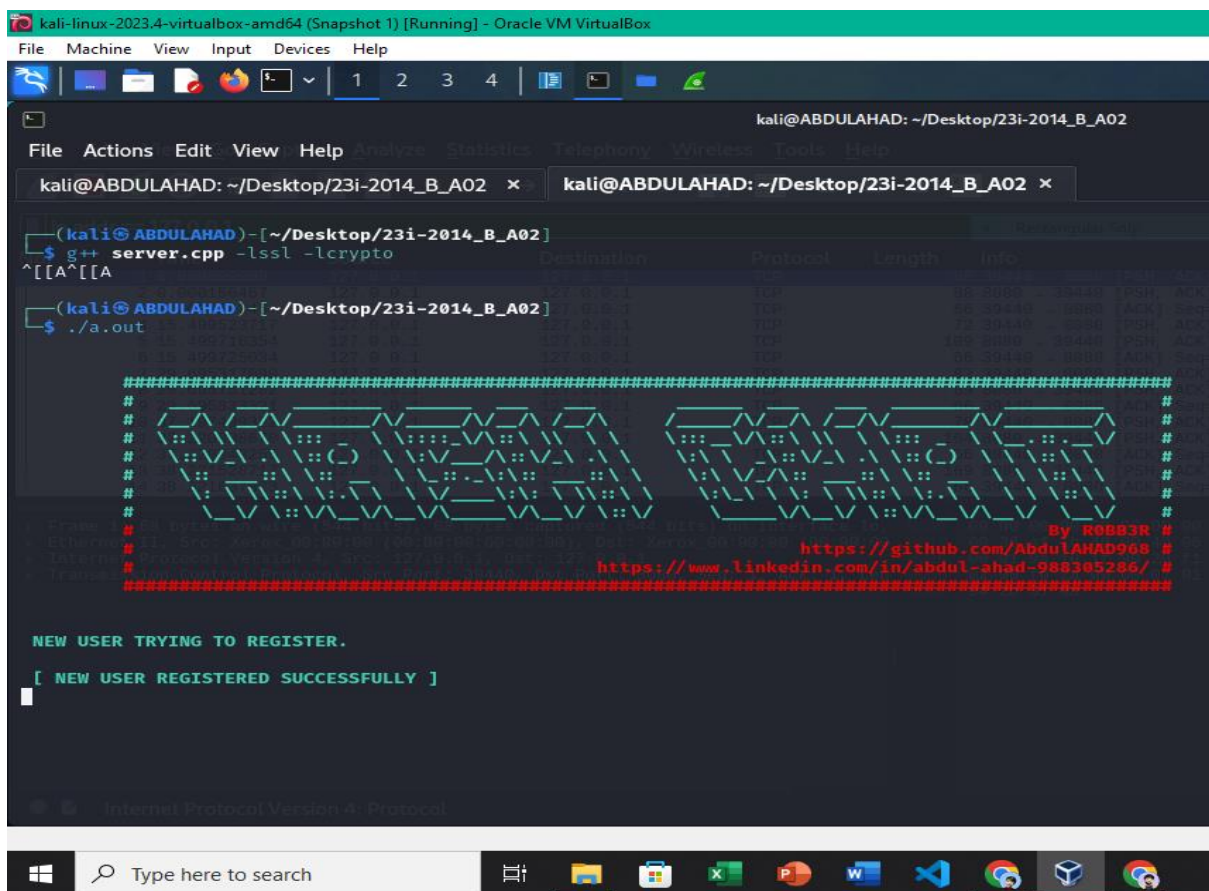
SERVER :: WEAK PASSWORD! [A-Z] - [a-z] - [0-9] - [special]
HIT THE KEYS = Hell78**86

SERVER :: ENTER YOUR PASSWORD =
HIT THE KEYS = hell78**86

SERVER ::
REGISTRATION COMPLETED SUCCESSFULLY.

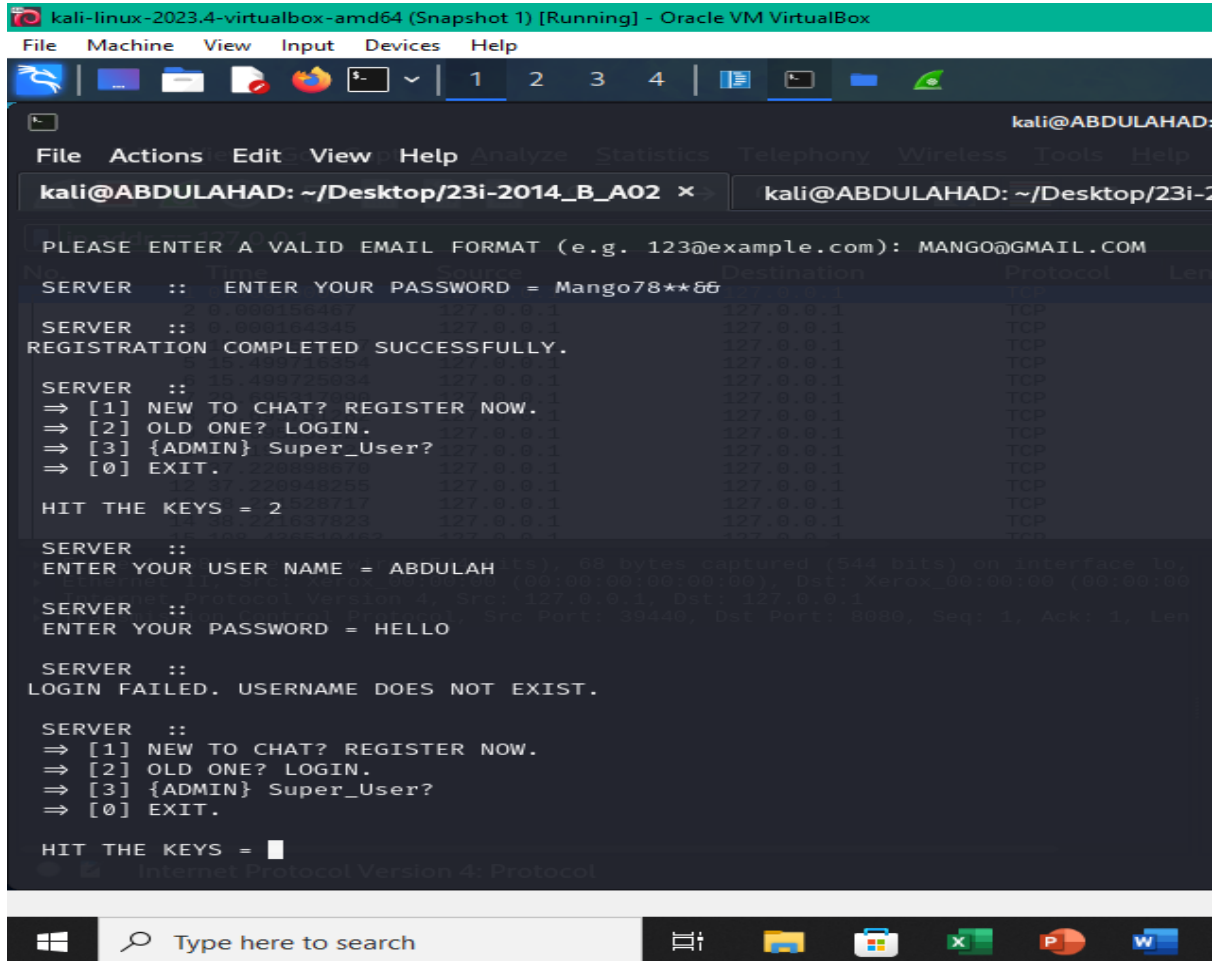
⇒ [1] NEW TO CHAT? REGISTER NOW.
⇒ [2] OLD ONE? LOGIN.
⇒ [3] {ADMIN} Super_User?
⇒ [0] EXIT.

HIT THE KEYS = █
```



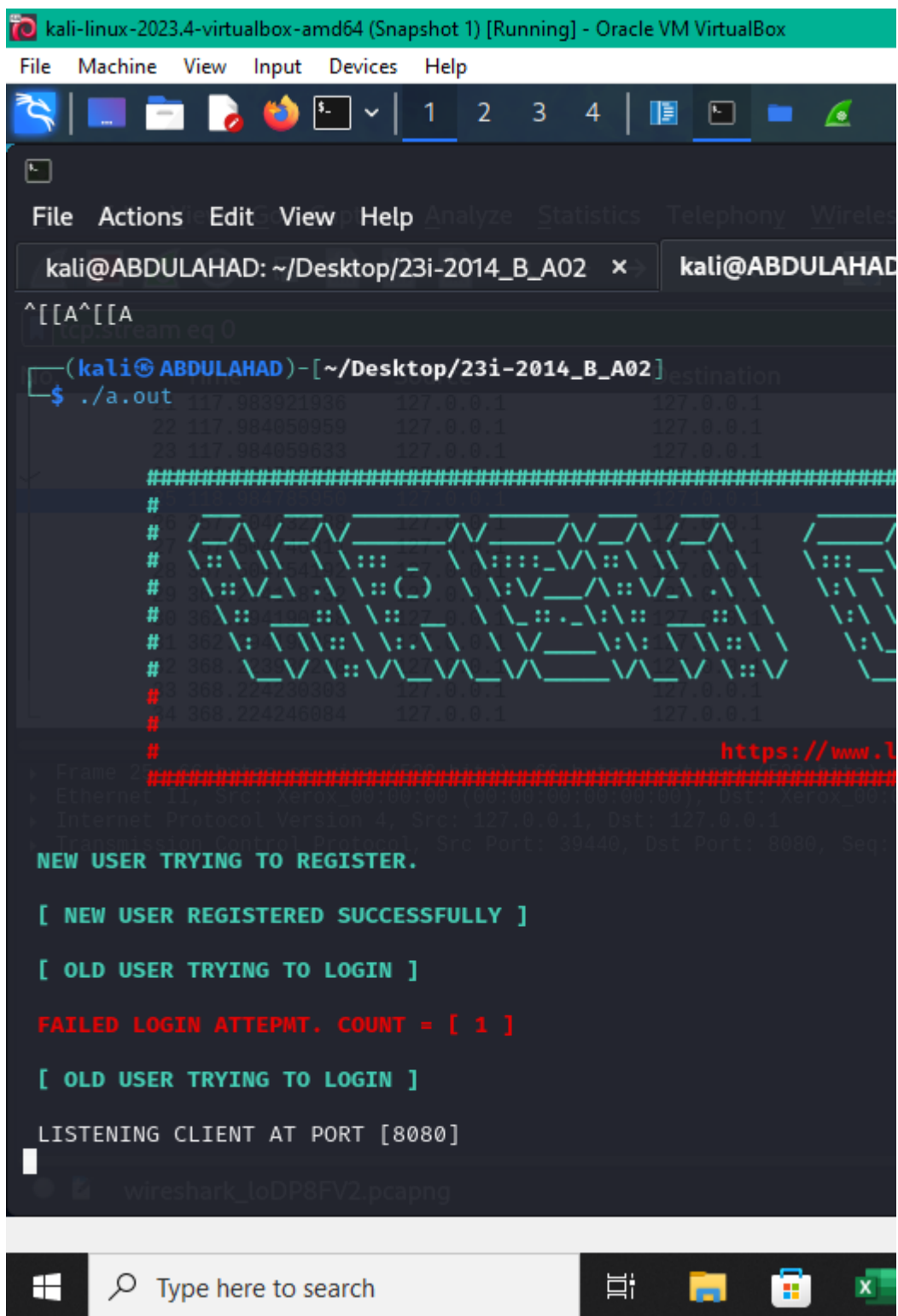
2. Login of existing users:

First I checked with the user that was not available in the file and verified that the user was not allowed to make a possible connection with the server.



```
kali@ABDULAHAD: ~/Desktop/23i-2014_B_A02 x>
PLEASE ENTER A VALID EMAIL FORMAT (e.g. 123@example.com): MANGO@GMAIL.COM
SERVER :: ENTER YOUR PASSWORD = Mango78**86
REGISTRATION COMPLETED SUCCESSFULLY.
SERVER ::
=> [1] NEW TO CHAT? REGISTER NOW.
=> [2] OLD ONE? LOGIN.
=> [3] {ADMIN} Super_User?
=> [0] EXIT.
HIT THE KEYS = 2
SERVER ::
ENTER YOUR USER NAME = ABDULAH
SERVER ::
ENTER YOUR PASSWORD = HELLO
SERVER ::
LOGIN FAILED. USERNAME DOES NOT EXIST.
SERVER ::
=> [1] NEW TO CHAT? REGISTER NOW.
=> [2] OLD ONE? LOGIN.
=> [3] {ADMIN} Super_User?
=> [0] EXIT.
HIT THE KEYS =
```

Then I used the existing users to go through the verification process and successfully completed the verification process and moved next to the chat phase.



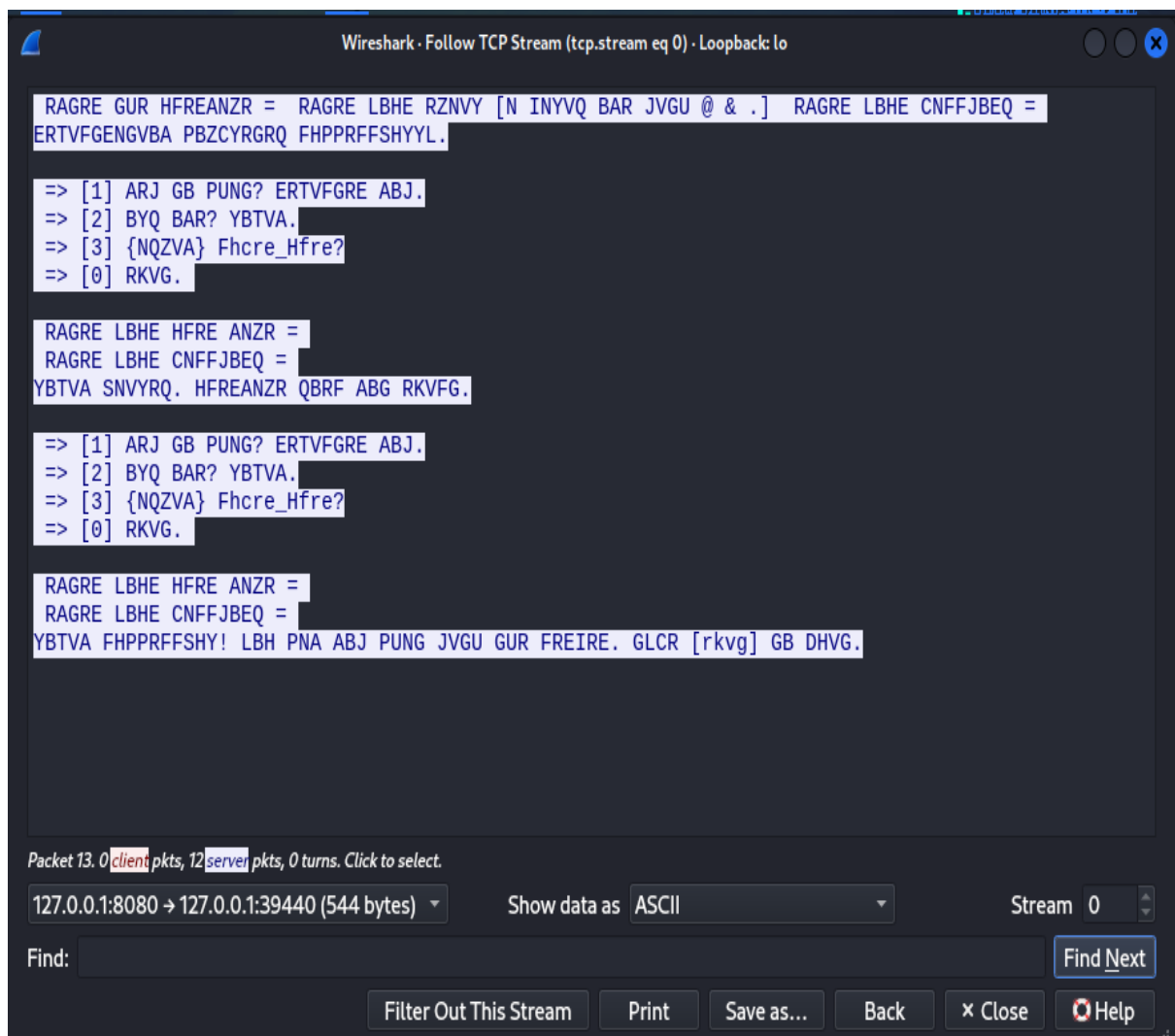
Wireshark Analysis:

Then I used Wireshark to see if the whole discussion was confidential or not.

This was the encrypted discussion over the channel. Outside the chat system I used simple rotation algorithm that works only for the alphabets and not the numbers or special characters.

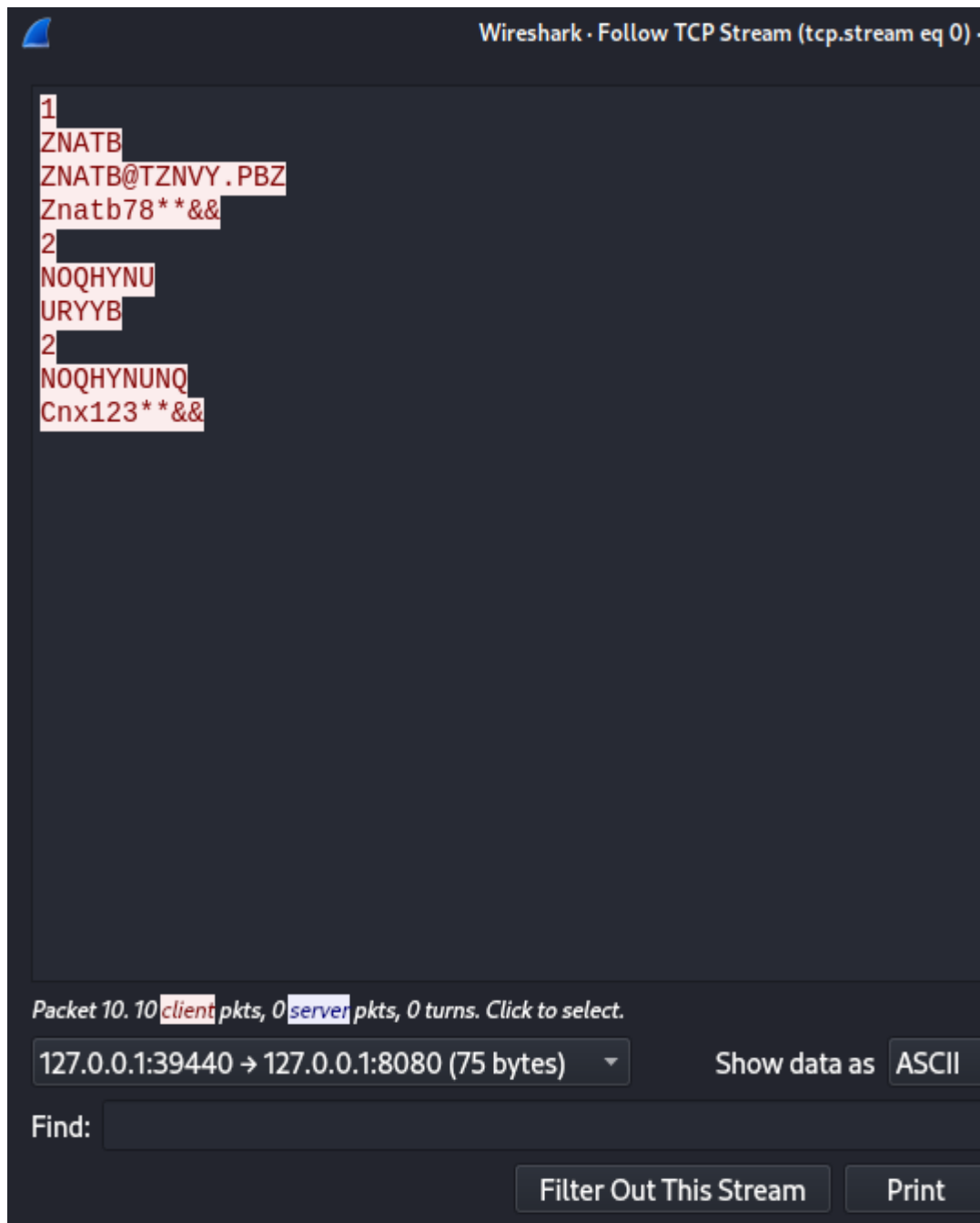
As can be observed in the picture. The message was encrypted with the same key and that key was also used for the decryption as the whole encryption and decryption process was symmetric.

This view is of the server side, as can be seen at the bottom there were 12 packets sent by the server. Analyzing this we can see that it is difficult to guess, although not impossible.

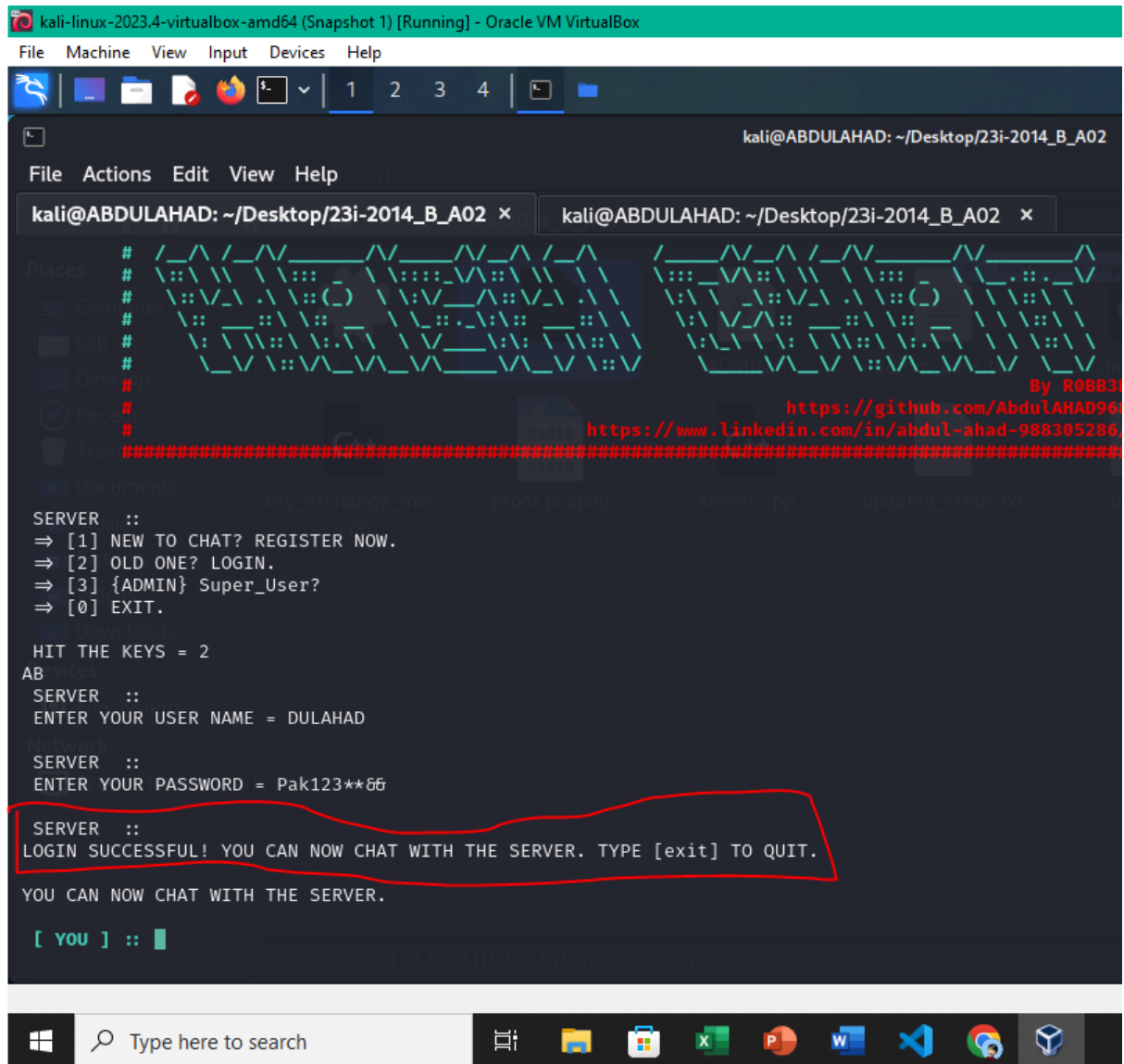


This is the view of client's packets that were captured by the Wireshark. Same here, only the alphabets are rotated by a certain number and not all the printable characters are changed.

This can be improved further by using the AES 128 encryption system.



After successfully making through the login face I moved forward toward the chat mode, where we did some chat and then observed the result in Wireshark. Here are some results.



Server-side chat encryption using AES-128:

Additional Functionalities:

I made a super user class and appended it to the header files so that I can access the code without making the separate cpp files executable.

```
#####  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
#####  
  
| { ADMIN MENU } |  
|-----|  
| 1. UPDATE [ADMINS] DATA. |  
| 2. SEE [ADMINS] INFORMATION. |  
| 3. EDIT [ADMINS] INFORMATION. |  
| 4. REMOVE USER. |  
| 5. SEE USERS INFORMATION. [HASHED INFO.] |  
| 6. SEE USERS INFORMATION. [PLAIN TEXT.] |  
| 7. [ EXIT MENUE] |  
|-----|  
| ENTER YOUR CHOICE :: |
```

First I checked with valid credentials and successfully entered the super user mode.

[illegible]

Here is the cases where I entered wrong credentials at the client side and this is what happened at the server side.

[illegible]

This was the client side view of the SU-57.



This was the final look of Wireshark interface after making all the communication. The .pcap file is also placed in the zipped folder. Can be checked for further verification.

