A screenshot of a cell phone

Description automatically generated

Connection Game

COMP454: Computer Networks

Dr. Ahmad Al Hajj

Done by:

Name ID Section

Khaled Sardouk 201901490 COMP 454

Mohammad Al-Tayyeb Soubra 201901076 COMP 454

Hadi Salloum 201900774 COMP 454

Salah El Deen Stouhi 201901265 COMP 454

Karim Hamod 201900057 COMP 454

Mohammad Kreidieh 201900799 COMP 454

Abstract:

The Game is about an attacker and a defender, the attacker will try to send a message “ATTACK” and try to protect the message, and the defender should try to not get affected by the message. There is a third-party member which is the server, which will provide the connection for the attacker and the defender.

We used PyCharm and Python language to implement our game. We also used LogMeIn Hamachi to make a VPN and use it to establish connection between the attacker and defender when we wanted to test it using 2 separate and far computers.

Roles of each side:

* The Server:

Establishes connection between the attacker and the defender.

* The Attacker side:
* The Defender Side:

Initial Setup for each Run:

1. First run server.py application.
2. Then run client.py application.
3. Then run client.py application again.

You will have 2 command windows, one for 1st client, and the other one for the 2nd client.

Both clients will take attacker/defender rounds.

The client that is the attacker for this round will automatically sends the message “ATTACK”.

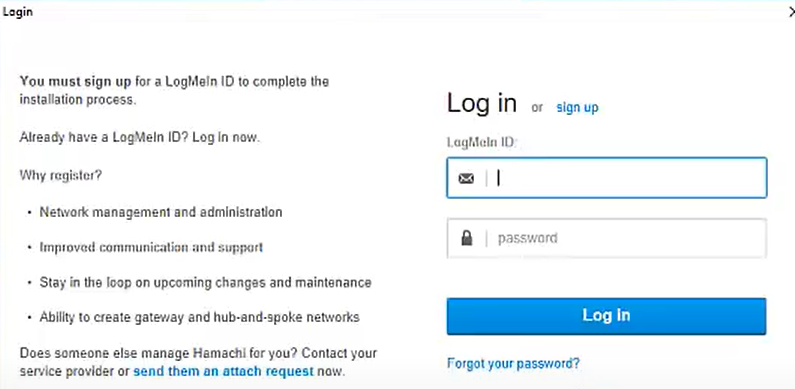
The client that is the defender for this round will encrypt it. Then decrypt it.

# How to use Hamachi:

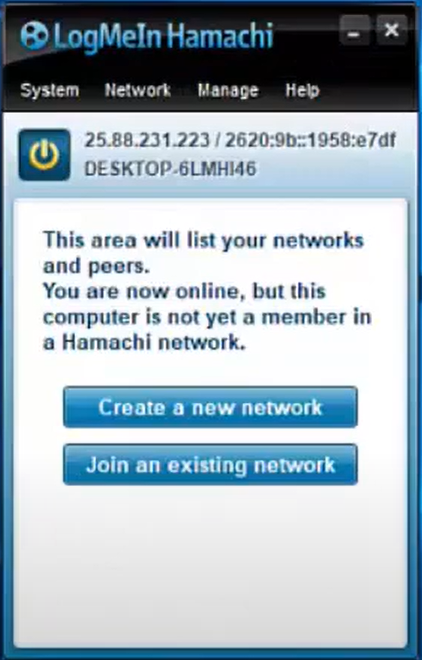
1. Turn it on



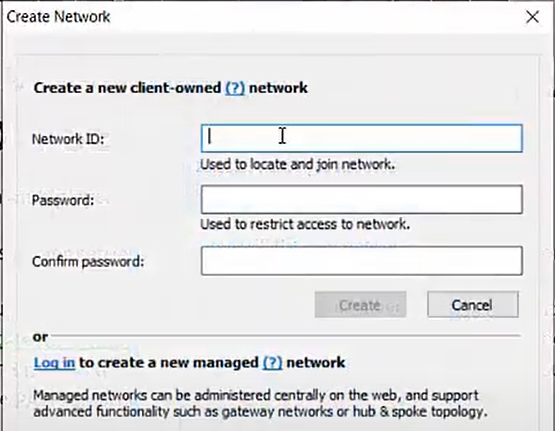
1. Login with an email or signup.



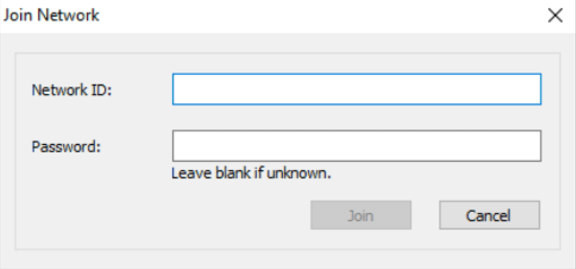
1. Create a new network



1. Name your network, give it a unique name because it will be your network ID.



1. After creating the network, the other user should choose “Join an existing network” and fill with the network ID and password.



1. After that let the

|  |  |
| --- | --- |
| Team | Members |
| Connection Team | * Khaled Sardouk * Karim Hamod |
| Attacker Team | * Salah Al Deen Stouhi * Hadi Salloum |
| Defender Team | * Mohammad Kreidieh * Mohammad Al Tayyeb Soubra |

1. Establishing the connection:

We used the socket library to create both the server and the client , in the server part we created a server socket and bind to the local host for testing first then to the ip address to the Hamachi room and to the port 1234, then we called the listen function of the socket which allow the server to accept new connections after we finished creating the server socket we moved to the game function in this function the server will accept two connections and this function we applied a primitive way to make a turn base game which is calling the function while changing the parameters and we thought to make it a game best of 3 , in this part it wall call the clients connection function which will assign roles to the two players and enable communication between them after the 3 turn we will exit this functions and call the check\_score function which will tell the two clients to calculate the score and send it to the server to see who won and reply to the clients and close the connection to the player and a

choice will prompt in the server side if we want to continue and it will reset the process.

For the client side:(we have 2 clients)

We first established the connection with the server by choosing a specific port to connect to and providing the host address where we used those 2 in a function called connect of socket library then we made a variable Response which is equal to the data sent from the server.

Then we made a loop where the client keeps receiving the data that is sent from the 2nd client to the server then to him.

Then we check the response:

-If it is “you are the defender” we receive the 2nd response and decode it into ascii form then process it using error detection functions we implemented and then we randomly scramble the response each time using NRZ or by shifting bits which is done through a choice variable which is chooses at random which method to use for scrambling .then if the result of the response after processing was “ATTACK” we decremented the defence score by -10 else we increment the defence score by 10 and send defence successful to the server .

-If the Response is “you are the attacker:

We start by initializing the message to be sent and encoding it in ascii form then we scramble the message by NRZ or by shifting also by variable choice at random .then we send it to the server which will send it to 2nd client we also receive the response from the server if attack failed or was successful at the same time when the connection was established we got the time of it and the time when we received the response by that we get the RTT and determined if the RTT more than threshold RTT we decrement the attack score by 20 and increment the defence score of the defender by 20 and if the response was that defence failed we incremented the attack score by 10 else we decremented it by 10

-At last, if the response was none of the above we print the total score of each client and send it the server which compares who has higher score and send to each one if he win or lose.

3- Error Control algorithm:

We used the Hamming Code to detect errors and correct them. The hamming code works by calculating the redundant bits in the packet sent, we use the following formula to calculate the redundant bits, 2^r ≥ m + r + 1 where, r = redundant bit, m = data bit.

We then use either even parity or odd parity to determine the values of our redundant bits. If we use odd parity, we count the number of 1s that correspond to that redundant bit, if the count was even, we set the bit to 1 else it is 0. And vice versa for the even parity method.

To determine the set of bits to use to calculate the redundant bits, we look at the position of the other bits. For example, if we were to calculate R1 (redundant bit 1), we need to look at all the bits whose position in binary end in 1, like 1,3,5,7,9, etc. As for R2 we need to look at all the bits whose position in binary include a 1 in the second position from the LSB (Least Significant Bit).

After calculating the amount, position, and value of redundant bits we send the packet, and the receiver will already know the amount, position, and value of redundant bits used, so he can compare that with the packet he received and if there is any error it should be corrected by itself by using the correction function that we implemented. The only downside of this error control algorithm is that it can only correct one bit.

Hamming Code source:

<https://www.geeksforgeeks.org/hamming-code-implementation-in-python/>

<https://www.geeksforgeeks.org/hamming-code-in-computer-network/>