EC 521 Final Project Report

— Public Network Credential Protections —

Team Members: Nick Gutierrez, Kaede Kawata, Suyash Bhatia, Balaji Sathyanarayanan

1. Description:

As Big data is a growing field in the industry today, people are sending huge amounts of data across networks for various practical purposes. Communication channels and networks were built with the idea of transferring information and little importance was given to security concerns, and sending information over a public network is not secure. Security measures have to be implemented to ensure secure communication. For example, VPNs are normally used to connect to the internet through public Wi-Fi. We are taking a multi-fold approach to ensure secure communication over an unsecured network.

2. Measurement:

Originally, our group was going to measure the problem by observing the traffic on public Wi-Fi using Wireshark. However, due to privacy concerns, we decided to host a server of our own on our localhost IP and observe the traffic on the server instead.We ran several tests to gauge the baseline for our solutions including running a basic Timing Channel Attack to get base runtimes for various inputs and sending several different strings over to network to see how they’re received by the client and traffic sniffer.

3. Mitigation:

Our group implemented a two pronged approach to solving the problem, the generation of decoy passwords to prevent password sniffing and a mandatory delay in response time to prevent timing channel attacks. To test our implementations, we set up a simple UDP server through the localhost IP address and client to communicate with each other. We also used wireshark to sniff any incoming or outgoing packets.

Decoy Generation:

In order to prevent attackers from sniffing network traffic and identifying a correct password, we came up with a method of generating decoy passwords to inject chaos into the network. Upon start up, the server will generate a decoy password and add it to an existing database table to be accessed later. This table stores ten decoy passwords at a time and removes the oldest password whenever an eleventh is generated. That way, there will be a rotating list of passwords that prevents attackers from memorizing the decoys.

These decoy passwords are generated using a random string generation. They are then checked against the real passwords using Leveshtein's distance to verify they are not too close to the real passwords.Initially, we experimented with a Multilayer perceptron. We tried training the MLP, but the lack of training data hampered this process.

The intended purpose of using string matching is to check the similarity between decoy passwords and real passwords which led us to choose Leveshtein's distance. Using Leveshtein's distance we ensure that our generated decoy password is different from any of the true passwords by a certain degree. Leveshthein's distance ranges from 0.0 to 14.0. 0.0 being that the two strings

are exactly the same. We have set a threshold at 2.0. If the similarity between our decoy password and real password is anywhere between 0.0 and 2.0 we reject the decoy password and generate another one. This security measure greatly helps us secure our true passwords.

After the server begins running, it begins listening for messages from a client. Upon receiving a message, the server randomly picks a decoy from the list and automatically sends it over the network. Since the user's real passwords are also generated using the same generate function, the real and decoy passwords will be indistinguishable from each other (*Fig. 1 and Fig. 2*).

Fig. 1: Real Password obtained through traffic sniffing

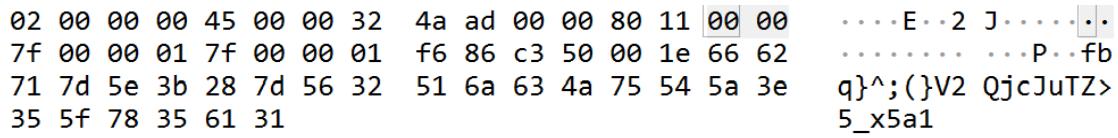
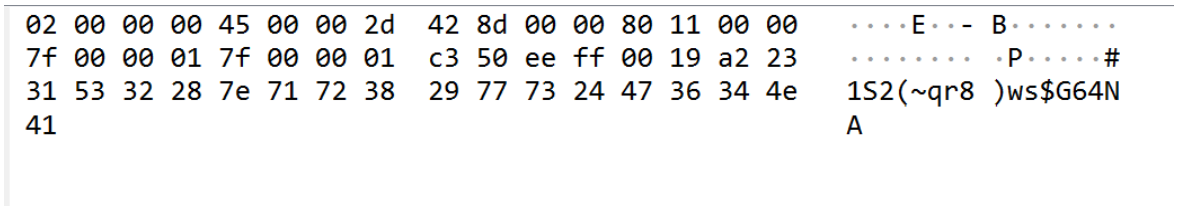


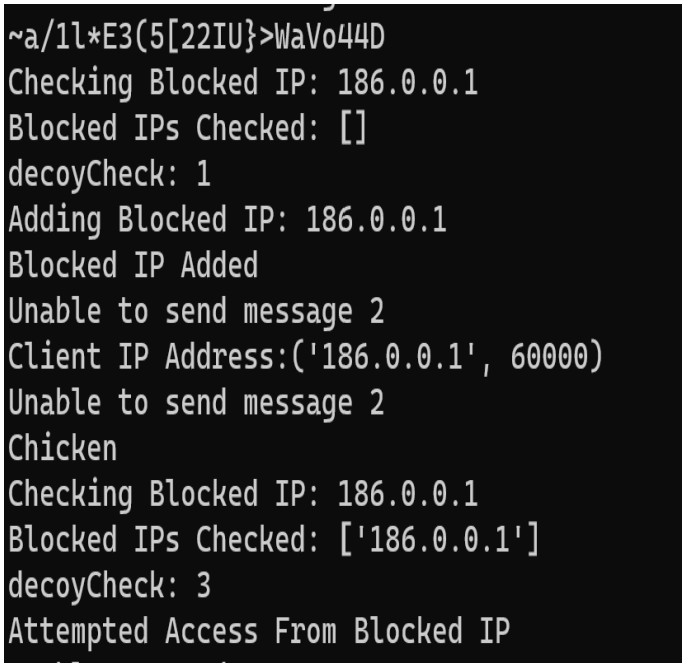
Fig. 2: Decoy Password obtained through traffic sniffing



If a decoy password is sent from a client, we automatically know they are an attacker since decoys can only be obtained through network traffic sniffing. Once a decoy is received, the IP address of the attacker is obtained and added to another column in our database table. This list of blocked IP addresses is checked every time a message is received. If a message is received from a blocked IP address, the client automatically receives an ACCESS DENIED message the server prints a message notifying the owners that there was an attempted access from a blocked IP address (*Fig. 3*). The approach we have taken here stems from simple probability. Using our

mitigation method we are trying to trap an attacker using decoy passwords which make it highly probable for an attacker to fall in the trap we have laid. We are trying to demonstrate a proof of concept here by using 10 decoy passwords. If we increase the decoy passwords to a 1000 it is highly probable that the attacker will enter a decoy password. The probability of the attacker entering a decoy password will be higher than the probability of the correct password being used.

Fig. 3: Server view of an attacker entering a decoy password and then attempting to enter a real password



Timing Channel Attack:

To prevent timing channel attacks on our server, we decided to implement a constant mandatory wait period, so no meaningful password timing measurements could be found. To accomplish this, we first measured the standard time it took for the server to receive the client message, check if the IP address was blocked, check the entered password against the decoy passwords, check the entered password against the real passwords, determine access status, and prepare to send the access status back to the client (*Fig. 4)*. After discovering all times took less

than 0.2 seconds, we set this to be our mandatory wait period. Finally, to enforce the wait period, a timer is started every time the server receives a message and is stopped once access status is determined. The determination time is then subtracted from the 0.2 second mandatory wait period and the server sleeps for the resulting time before sending the status back to the client. The result is a constant access determination time for the server, with the only time variation on the client side being the time it takes for the access status to be received by the client (*Fig. 5*).

Fig 4: Normal status determination times for the server (Note: Chicken is an example real password)

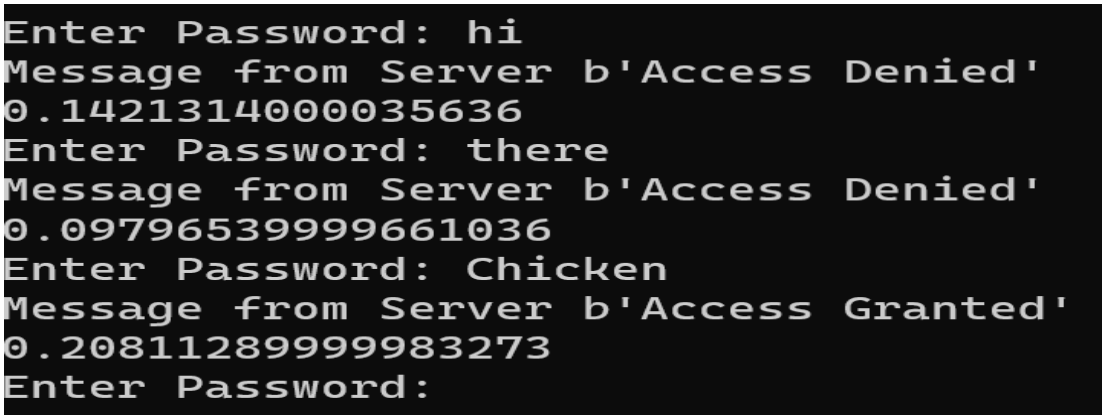
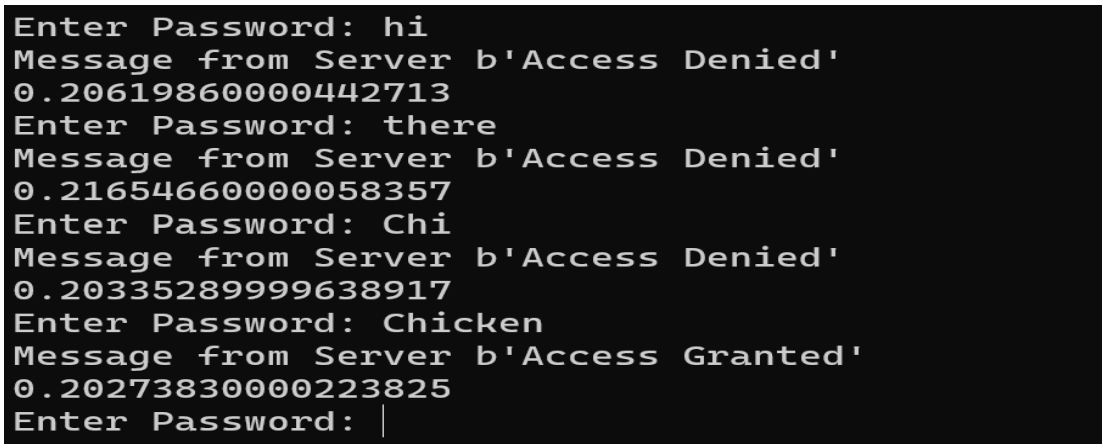


Fig 4: Status determination times for the server with mandatory wait period enforced. Note that time is no longer determined by the number of correct letters entered



4. Glossary:

Decoy Password: Password generated by the random password generator to trap attackers.

True Password/Real password: Correct passwords for the server

Time Channel Attack: **timing attack** is a side-channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms

UDP: Communications protocol that is primarily used to establish low-latency and loss-tolerating connections between applications on the internet.

Wireshark: Free and open-source packet analyzer used for network troubleshooting, analysis, software and communications protocol development, and education

IP address: A unique string of characters that identifies each computer using the Internet Protocol to communicate over a network.

network traffic sniffing: **Reading data packets traversing the network within the Transmission Control Protocol/Internet Protocol**

Localhost: **computer running a program; in which** the computer works as a virtual server.

Multilayer perceptron: fully connected class of feedforward artificial neural network.

Leveshthein's Distance: **Measure of the similarity between two strings**

Server: Piece of computer hardware or software that provides functionality for other programs or devices, called "clients"

Sniff: **Process of monitoring and capturing all data packets passing through given network**.