Progress Report

* Project Objective
  + Demonstrate how to crack various types of passwords that is stored or leaked online
    - Includes various types of encryption/hashing, and plain passwords
  + Demonstrate how to protect against these various attacks using salting and some other lesser secure protection methods
    - Includes most common 4 digit salting functions and also more complex functions
* Project Approach
  + Creating a dummy list of fake ‘leaked’ passwords that would be found on the internet
  + Create a rainbow table cracker as well as standard encryption cracker in python that would attempt to find the password value
  + On the defense side, create a salting function that would be a separate program, so when the user types in his/her password, we can demonstrate how salting makes rainbow cracking theoretically impossible because of the time difference in cracking it
* Project Tasks
  + Create a program that allows us to both demonstrate how a user’s password is hashed, and how we can steal that password
  + Generate rainbow tables based on a fake list of ‘leaked’ passwords from the internet
  + Show how this rainbow table cracker can be used to steal passwords, despite hashing being a one-way function
  + Create a basic four digit salting function to demonstrate salting protections
  + Generate a more complex, theoretical protection against rainbow cracking
* What’s been done so far
  + Dummy database of passwords in a basic txt file
  + Rainbow Table Generator is complete and most of the rainbow cracker
  + Standard decryption and encryption
  + Research on the various hash functions used, including MD5 and SHA-1
  + Research on how a rainbow table ‘reverses’ the one-way hash function
  + Research on how salting a password makes password cracking take exponentially longer
* To Do List
  + Create the salting function
  + Update/automate some parts of the process
  + Find a better method, if possible, for defense
  + Perhaps create UI

Research

In this preliminary research, we will be looking into how passwords are stored online, and how their leak can differ in severity depending on how the information was saved by the service provider.

There are several methods for a website or a service provider to store user passwords these days. Some are outdated and dangerous while others are becoming the standard by the security point of view. Obviously, letting large and reputable service providers handle user login/security information, through Google or Facebook login for example, is the best way to avoid a leak. However, this is not always available or best fit for projects nor is it readily available for us to test with. Thus, this method will not be considered for this project.

Below is the list of methods that will be tested/replicated:

* Bad: Storing password (and possibly its hint) as a plain text in a database. This is bad since in case of a data breach, all passwords along with their user info will be read easily by the attackers.
* Bad: Hiding passwords behind an encryption. This is only marginally better than above since all passwords are only one key away from being read by attackers. If there are insiders providing encryption key, it is nearly useless as a security method.
* Better: Hashing every password. This is a better method but has its own flaws. If there are multiple repeating hashcode, this can be a giveaway for a common password. Attacker can also use something like rainbow-table that replace computing power with disk space to accelerate the cracking process.
* Best: Hashing with salting. Hashing with slating is a method where during the hashing process of the passwords, the service provider adds random string of characters or numbers and hashs the combined product of salt and the actual password. This way, during the check, password can be recombined with the salt to compare against the hashcode. This provides very high security for every single password’s hashcode will be different from each other and cracking of the code itself can be near impossible depending on the implementation.

References:

<https://www.2brightsparks.com/resources/articles/introduction-to-hashing-and-its-uses.html>

<https://www.youtube.com/watch?v=8ZtInClXe1Q>

Plain text password, if leaked is obviously bad since it is readily accessible:



One way to protect this info from a potential leak is to encrypt each password:



Turning plain text passwords into something like this:



The owner of the database will have the encryption key. However, as discussed in the research above, this key is often leaked.

Once the adversary gets hold of the key, it becomes trivial to decode each encoded password:





We can improve security even further by hashing every password instead of encrypting them with a key. One such method is via SHA1.



However, with rainbow table and enough computational power, easy to normal passwords can get looked up.





**Report written by Yaxiaer Atajiang and Knicholas Kennedy**

**Code written by Yaxiaer Atajiang**