Eric Elguea

Assignment 1

CS 4351

Lab 1 – Cracking Passwords

**Introduction**

The purpose of this assignment is to become familiar with Message digests, the concept of hashing a password, salting technique, and cracking passwords online and offline, with a brute force or dictionary approach.

**Dictionary Attack**

For the dictionary attack portion of this assignment, I downloaded the ***wordsEn.txt***file, which has over 100,000 entries of words from the dictionary. This file is used widely for dictionary attacks. I then used the provided test file from Dr. Longpre that had the user’s contents in this format [username, salt, hashed password (SHA256, then SHA1 to get the final code)]. The main goal of this code block is to concatenate the salt of each user with each individual word of the dictionary, then following the SHA256 -> SHA1 process, as stated in the assignment memo, to find the hash. If that hash matches with the hashed user password, that word concatenated is the password for that user. This was coded in Python and was executed in 2.1 seconds. The output for the program is shown below.

A screenshot of a cell phone

Description automatically generated

Each password was tested and was successful for each user. Early problems arose when the splitting of the salt to concatenate was taking extra whitespace that was not needed. This is countered by doing an additional strip of all whitespace prior to splitting the salt from the full user contents.

**Random Attack or Brute Force Attack**

For this portion of the assignment, the test file provided had the user contents as [username, hashed password]. This approach differs as there is no salt to be added to find the matching hash. For this, we were tasked to find all possible combinations, or permutations, ranging from length 1 to length 10. The Python library **itertools** was used to perform each permutation by way of function calls. These were then hashed following the SHA256 -> SHA1 process, to find the hash. If the hash matched the hashed password in the user contents, that permutated password is the correct password for the user. The output for lengths 1 to 3 are shown below as the runtime for lengths more than 4 causes the system to crash, leaving the total time for length 3 to be 15.2 seconds. The first three passwords were checked and successful for the three users tested.

A screenshot of a cell phone

Description automatically generated

**Online Attack**

The online attack consisted of only one user with a password of length 2 using only lowercase letters from a-z. This password had to be done in the webpage <http://cssrvlab01.utep.edu/classes/cs5339/longpre/loginScreen.php>.

For the online attack, I went with a Java approach as I knew that this would be the easiest to code in Java. The approach follows a brute force approach, as the previous portion and goes through each possible permutation of the alphabet. In order to access the website, a shell script was created to access the website remotely and continuously run the program until the correct password was found; since my shell scripting isn’t strong, an Experts Exchange article aided on creating the script, along with a curl documents page to help send the password data. The terminal output is shown below.

A picture containing device

Description automatically generated

**References:**

**wordsEn.txt -** [**http://www-01.sil.org/linguistics/wordlists/english/**](http://www-01.sil.org/linguistics/wordlists/english/)

**itertools -** [**https://docs.python.org/3/library/itertools.html**](https://docs.python.org/3/library/itertools.html)

**curl POST with data -** [**https://curl.haxx.se/docs/manpage.html#-d**](https://curl.haxx.se/docs/manpage.html#-d)

**grep and password reset script -** [**https://www.experts-exchange.com/questions/24696217/How-to-make-a-password-reset-script-linux.html**](https://www.experts-exchange.com/questions/24696217/How-to-make-a-password-reset-script-linux.html)