Daniel Ornelas

Assignment 1

CS 5352

Cracking Passwords – Assignment 1

**Introduction**

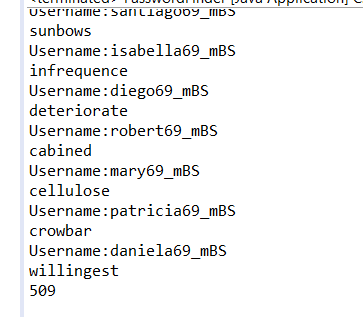
The purpose of this assignment was to get familiar with hashing and salting as well as cracking passwords using different methods. Salting is a technique used to make passwords harder to crack as it adds length to the password, and if an attacker knows the password he would need to guess the salt, making the cracking of the password more time consuming. The three types of attacks performed were a dictionary attack, a random password attack, and an online attack. The attacks were done on individual usernames and passwords provided by Dr. Longpre. For this report, we will be showing results for the individual passwords #69.

Note: A user object was created to easily access each user in the program by adding all users in a list.

**Dictionary Attack Approach**

For the dictionary attack I downloaded the WordsEn.txt dictionary. The user files for the dictionary attack were in format [username, salt, hashed password]. The hashing used in all usernames is SHA-1. The idea behind this attack is to try to crack the password by concatenating each possible word in the dictionary with the user salt and hashing using SHA-1, then comparing the hashed password computed with the one provided in the text file. If the passwords match, then we store the password and we go into the next user. To compute SHA-1 an external library named Digest, from apache commons codec was used.

**Dictionary Attack Results**

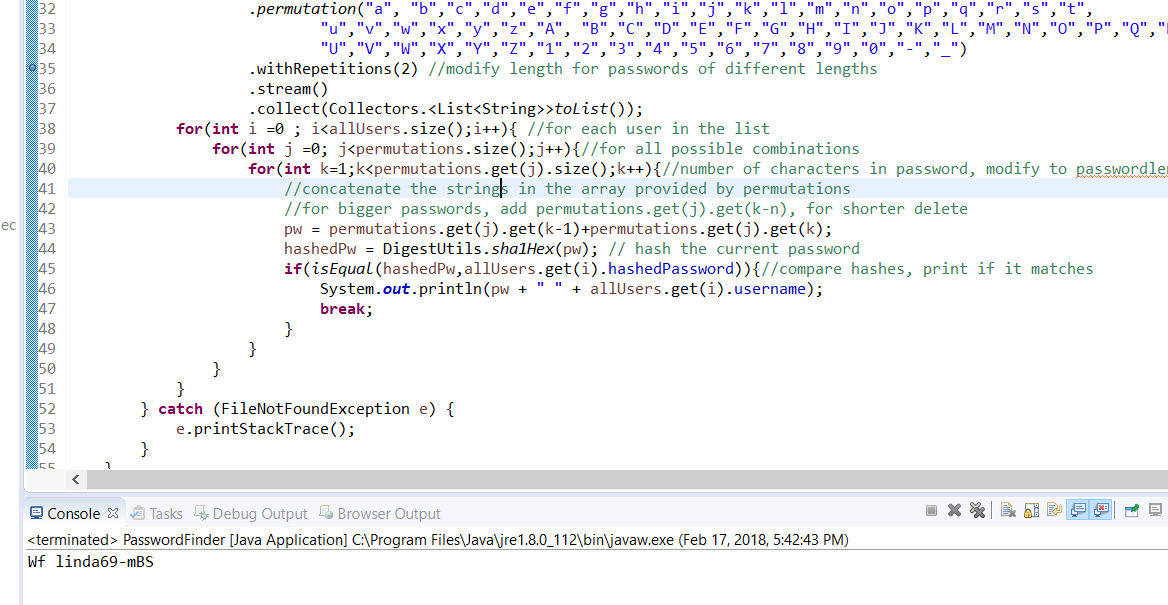
****

The program easily found all password in just 509 milliseconds.

**Random Attack**

For random attack, the user file contained 10 entries in format [username, hashed password]. The passwords for this attack were of length 1-10 (one of each), and they used combinations containing: a-z, A-Z, 0-9, - or \_. The approach taken was similar to the previous attack, all users were stored in a list, but this time I used a library called combinatorics to generate all possible permutations of a given length using the possible characters. Each combination was hashed using the same SHA-1 library and passwords were compared, only the correct password was printed. The problem with this program is that the combinatorics library would take an input of the length of the password, therefore the code had to be modified in each run to get the next password. Every time a password was found the user was removed from the text file and the program was ran with the remaining users.

**Random Attack Results**

****

The program seemed to run fine, but errors came up because of the library used for the combinations. For password of length 1 and 2 the program took less than 1 second to find the password. For length 3 it took 3 minutes, and for length 4 it took 11 minutes and 23 seconds. Password of length 5 could not be cracked as the combinatorics library stored all permutations of the string in a list and the heap space threw a memory exception in Java. The other drawback was that the problem had to be modified for each length password. Modifications that need to be done are in the code comments.

**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://cs5339.cs.utep.edu/longpre/loginScreen.php>.

**Online Attack Results**

To get this password I used the same combinatoric library that was used for the random attack, but this time I only created all permutations possible from a-z with only 2 characters. Then a POST request was sent to the webpage with un(username) and pw(password) as parameter for each password attempt. The program slept one second after each try to do not overload the server. To avoid printing each invalid attempt and only printing the answer into the console, I noticed that when the password was correct, the webpage displayed the number of attempts. So, I added a condition to the program in which it would only print the string that contained the word “attempts” and then it would print the password. The online attack took my program less than one minute since the password was “ra” and the list from the combinatorics method does the following combinations {aa,ba,ca,da,…} therefore it found ra in a matter of seconds.

**References**

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

**1000 Most common words in English- Can be used for dictionary attacks -** [**https://www.ef.com/english-resources/english-vocabulary/top-1000-words/**](https://www.ef.com/english-resources/english-vocabulary/top-1000-words/)

**Top 10000 most common passwords -** [**http://www.passwordrandom.com/most-popular-passwords**](http://www.passwordrandom.com/most-popular-passwords)

**Java Combinatorics Library -** [**https://mvnrepository.com/artifact/com.xiantrimble.combinatorics/combinatorics/0.2.0**](https://mvnrepository.com/artifact/com.xiantrimble.combinatorics/combinatorics/0.2.0)

**Apache Commons Codec (SHA-1) -** [**https://commons.apache.org/proper/commons-codec/download\_codec.cgi**](https://commons.apache.org/proper/commons-codec/download_codec.cgi)