**Abhishek Gautam UB# 50169657, agautam2@buffalo.edu**

Homework 3: this homework consists of only 1 question, and all parts are mandatory. All 5 parts require coding. Coding is required in Python. A long with your code, provide a sheet including steps followed and comments (where requested). The code will be automatically graded, so provide what is requested. If in doubt, as on Piazza.

Deadline and notes: 1 week from the time stamp of the posting of the homework on Piazza. No late homework is accepted. No exceptions. Academic integrity is taken seriously, and cheating is not tolerated. Note about data: do not distribute the file. When unzipped, the file is 2.8GB

Question 1 (100pts) attached you will find a large file with a large number of repeating ASCII passwords (320,412,510 passwords) corresponding to 2,151,219 unique passwords. Based on the contents of the file and what you learned in the class, perform the following (using python):

1. **Frequency analysis:**
   1. Perform a frequency analysis of the passwords. The outcome of this analysis should be a sorted file with two columns. The first column in the resulting file is the number of times a password appears in the file, and the second column is the password itself. The second column is the set of unique passwords in the file.

**Solution**: See the attached output file, code and the README.MD file.

* 1. What do you conclude about the distribution of the passwords and their frequency? Comment on whether that is surprising or in line with your expectations.

**Solution:** Distribution of passwords is irregular. Some obvious outcomes were observed like a password that would likely be called as *Strong Password* (One that is not easy to guess and contains multiple characters) has less frequent occurrence. The *Weak Passwords* occur more frequently.

1. **Strength analysis:** 
   1. Perform a strength analysis of the passwords in the file. The strength analysis should follow the Shannon entropy. Use the entire set of passwords in the file as your space of probability calculations. Here is an example of how the Shannon entropy works with the entire file as your calculation base. Let PWD = {Aziz, Mohaisen, is, a, har$h, professor} be your dictionary of passwords. First you calculate the probability for any character (case sensitive) to happen in any password (Aziz, Mohaisen, etc.). For example, all passwords have 29 characters in total, and A happens only once, so the probability for A is 1/29. Probability of a, on the other hand is 3/29, and so forth. Then, for a given password w, you construct a probability distribution over the characters, plug that distribution in the Shannon formula explained in the class, and done. The outcome should be a sorted file of two columns. The first column is entropy value, and the second column should be a unique password in the passwords dictionary (there is no point of including the same password multiple times, since they will have the same entropy).

**Solution:** See the attached output file, code and the README.MD file.

* 1. Comment on the strength analysis of the passwords and any obvious trends

**Solution:** The output could be used to determine the relation between the strength of the password and it entropy. Observed behavior was that passwords that were long and with unique characters that didn’t repeat had a large value of entropy. On the other hand, with decreasing password length, entropy value decreased. With repeating characters, entropy value was low.

1. **Strength analysis of unique passwords:**
   1. Redo question 2, but this time using the set of unique passwords in column 2 of question 1. This is, when calculating the probability as in question 2, consider your space of probability to be driven from the unique passwords dictionary. Compare the results with question 3.

**Solution:** See the attached output file, code and the README.MD file.

* 1. Comment on the comparison, and how the frequency of a password in a dictionary makes that password a good (or a bad) choice with respect to security.

**Solution:** In this case, the password base was unique unlike the former scenario. Similar relation between the entropy value and the password length was observed as in the previous case. The outputs of the two cases differed only in the output that too at 3 to 4 decimal places.

1. **Offline dictionary attacks:** The hash values (MD5 hashes, installed) below were revealed when a DB was compromised. Answer the following
   1. If you are to try to find the original passwords in those hashes, how would you approach the problem?

**Solution:** Resolution to this will be difficult as we have no prior knowledge that the passwords exist in the dictionary. This is hard due to the one-way property of the hash. The only option could be to trying to solve this on the internet and using those results along with some hit and trial efforts.

* 1. If you are told that the passwords are already in the dictionary you are given, how your approach would be different?

**Solution:** Since this time we know that passwords exist in the dictionary, this can be done by:

* First finding unique occurrences of passwords as done in question1.
* Then hash value can be calculated for each of these uniquely occurring passwords.
* Comparing these calculated values with the given hash values.
  1. Find the passwords of those hashes, given that they are in the dictionary.
     1. ba931c15ec0163c4bb339f41571694ce
     2. c9cd905fc459e5550b8c3b01d4346c25
     3. e9269d9e52a692f52caece9d0e7cdae1
     4. 660719b4a7591769583a7c8d20c6dfa4

**Solution:** See the attached output file, code and the README.MD file.

1. **Contexts for targeted password cracking:** Jim is a fan of the Buffalo Bills. This, in turn, is reflected in his choice of passwords. Often time, he would use his passwords as words related to the Buffalo Bills, including names of players, years of wins, etc. The MD5 (unsalted) hash value of Jim’s password is “83bfc234f88cc75d52e9ec24e54bc8be”. Answer the following:
   1. Provide 3 possible candidates for what the password could be

**Solution:** Three possible candidates can be:

* 1992afc - Bills won AFC Championship in 1992.
* Kemp15 – Number 15 was issued sparingly after Kemp’s retirement. Is in circulation now, but no one wears it.
* Ryan1960 – Rex Ryan is the current coach and is admired by the people. 1960 is the year when Bills was found.
  1. If you are told that the password is a family name of a player playing for the Bills, how would this help you find the password?

**Solution:** This would narrow down the search to only name of the players and that too surnames. So the guessing work reduces, as the search domain is less.

* 1. If you are told that the password is a family name of a player who retired on April 7th, how would this help you find the password?

**Solution:** This would narrow my search down to players who retired their jersey on 7th of April and look for their family names. For example Google search reveals that Tarpley retired on April 7th. A data of such entries can be created and then these can be hashed and compared with the given values to find a match.

* 1. Use a side channel to infer the password, without knowing the context above (e.g., search Google).

**Solution:** Steps for this:

* Go to <http://www.md5online.org/md5-decrypt.html> or any other online tool that decrypts md5 hashes.
* On using it, this gives back the value – Tarpley and hence the password can be inferred.
  1. What does this tell you? About salting, using common passwords, etc. **Solution:** Passwords that are not common will possess a greater difficulty in breaking if unsalted. On the other hand, common passwords can be broken. This tells us that if data related to the password is leaked, it can be broken easily. This becomes more significant if the password is one word like in the above case, *Tarpley.* A scope of improvement is adding some other piece of information to your password thus making it difficult to break instead keeping it a single word about which information is leaked.