CSE 543 - Spring 2015 - Assignment 1: Password Selection

1 Dates

- **Out:** *January 20, 2015*
- **Due:** *February 5, 2015*

2 Introduction

In this assignment, you will research and evaluate methods to improve passwords chosen from dictionary words. Users often base their passwords on dictionary words to help them remember the password. However, crackers and guessing algorithms take advantage of this knowledge to guess passwords, enabling them to find dictionary passwords more quickly than other passwords. Your task is develop and justify three candidate methods for password generation and evaluate them using the provided password cracker.

3 Background

Password Guess Number The "Guess Again" paper [2] (from class) describes how Markov and PCFG methods may be used to estimate the "guess number" for a password, approximating the strength of the password. See the paper for details.

Password Improvement Researchers have suggested that modifying a few characters in a user-supplied password such that the guess number exceeds a threshold. For example, Houshmand and Aggarwal [1] suggest that changes to only one or two characters in a user-defined password (either by replacement or insertion) may be sufficient to strength the password such that its guess number exceeds a threshold.

For example, simple, user-chosen passwords, such as life45!, are changed automatically through the addition and/or replacement of characters to passwords such as liFe45! whose strength against password guessing surpasses a threshold. An important facet of this work is that the number of password modifications must be sufficiently small to enable the users to remember the strengthened password effectively.

Threat Models However, a threat to such methods is that the adversary may know the strengthening algorithm and use this knowledge to improve their ability to guess the strengthened password more efficiently than suggested. Schmidt and Jaeger [3] show that if strengthened passwords are added to the training set, then the resultant guess numbers of newly strengthened passwords are reduced, in some cases below the threshold. In addition, brute force methods may successfully crack such changed passwords if the original password is weak enough and the number of changes are sufficiently small.

4 Problem Statement

Your task is to study the background research on password guessing and password strengthening to identify **three** password strengthening methods that start with a dictionary word (in English) and produce a strengthened password with no more than **three** distinct modifications to the password. I do not want to place any restrictions on the modification methods, so modifications may change more than one character at a time. However, each change must be a single "unit" such that the change can be understood by users.

5 Exercise Steps

The project consists of the following steps.

5.1 Password Strengthening Methods

First, study the papers to develop candidate password strengthening methods. You have to write a justification for each of your **three** suggested methods both as to why they would strengthen passwords effectively and why a user may be able to memorize the resultant password.

You will have to implement these methods such that you can generate a set of strengthened passwords from a set of dictionary words. I need to be able to run your strengthening methods on a set of dictionary words that I supply to evaluate your suggestion. You will need to submit this program as a deliverable.

5.2 Password Cracking

For password cracking, please use the rockyou password database as an example. The rockyou password database is available at `https://wiki.skullsecurity.org/Passwords`.

To understand password cracking, I provide password cracking code at `http://www.cse.psu.edu/~tjaeger/cse543-s15/password_cse543.tgz`.

These programs perform the following tasks.

- **splitpwd**: Takes a password database and splits it into files that include passwords of length longer than an input and groups passwords by how many types of characters the passwords include. The types of lower case letters, upper case letters, numeric characters, or symbol characters (remaining).

  ```
  splitpwd min-length file1 [... filen]
  ```

  That is, more than one file may be submitted to `splitpwd`, although for this project you need only submit the rockyou file.

  Assuming that the name of the input file is `rockyou.txt`, the main output of `splitpwd` are files that contain 1, 2, 3, or all 4 types of characters listed above. The names of such files will be `file1.min.types`, where the file consists of the password file `file1`'s passwords, of length greater than or equal to `min`, which contain the number of characters specified by `types`.

  Files are further partitioned into four subfiles a, b, c, d for use in training.

- **markov**: This program produces Markov chain and PCFG information useful for password cracking.

  ```
  markov 1 file
  ```

  Where 1 is the size of the nGrams used in building Markov chains and PCFG structures - we will use 1 - and file is a file produced by `splitpwd` to be used for training only.

  The `markov` program produces two files: (1) `file.mc1`, which contains the Markov chain training data and (2) `file.pcfg1`, which contains the PCFG training data.

- **MarkovCracker**: This program uses the Markov chain training data to crack passwords.

  ```
  MarkovCracker 1 test-file markov-file > crack-file
  ```

  The `test-file` should be the password file that you generated from dictionary words and the `markov-file` should be Markov chain training data produced above.

  The cracker will produce output that you should redirect to a file (e.g., `crack-file`).

- **pwdstats**: This file takes the cracker output and produces output in the form of percentiles and the guess number for passwords as computed by the cracker.

  ```
  pwdstats < crack-file
  ```

5.3 Teams

The project teams are as follows:

1. Rengasamy Prasanna Venkatesh, Minkin Ilia, Sharma Aakash
2. Jadidi Amin, Saghaian Nejad Esfahani Sayed, Wang Shuai
3. Qiu Li, Narayanan Iyswarya, Zientara Peter
4. Cao Wenqi, Upreti Nitish, Lv Weining, Wang Kaiyu
5. Nima Elyasi, Mukhopadhyay Manjari, Xu Dongpeng

6 Deliverables

Please submit the following:

1. Your password strengthening program and necessary makefiles for it.
2. Your dictionary word file used as input.
3. Output from `pwdstats` from use of `MarkovCracker` on the rockyou password database.
4. A writeup that contains three sections: (1) background on password strengthening and password cracking (based on your understanding of the research papers); (2) a justification for the **three** password strengthening methods you evaluate; (3) a description of the evaluation that includes any additions to the experimental setup above, the experimental results, and the evaluation of those results.

7 Grading

The assignment is worth 60 points total broken down as follows.

1. Three password strengthening methods and their justifications (20 pts)
2. Code for those methods (10 pts)
3. Writeup (20 pts)
4. My evaluation of your algorithm (10 pts)

References

[1] S. Houshmand and S. Aggarwal. Building better passwords using probabilistic techniques. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 109–118. ACM, 2012.

[2] P. Kelley, S. Komanduri, M. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, Oakland '12, pages 523–537. IEEE Computer Society, 2012.

[3] D. Schmidt and T. Jaeger. Pitfalls in the automated strengthening of passwords. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 129–138. ACM, 2013.