CSE 543 - Fall 2016 - Assignment 1: Password Generation

1 Dates

- **Out:** *August 30, 2016*
- **Due:** *September 15, 2016*

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 a method for password strengthening and evaluate its effectiveness using the provided password cracker. The most effective method chosen will win a bonus, so the aim of the evaluation is to enable you to develop the most effective password strengthening method.

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 the **best three characters and new character values** to modify to strengthen passwords that transform dictionary words (in English) to produce strengthened passwords with the highest guess count.

5 Exercise Steps

The project consists of the following steps.

5.1 Password Strengthening Methods

First, study the papers to propose candidate password strengthening methods. **Each team member must propose one candidate strengthening method and present why your method makes a promising hypothesis. Please identify which student suggested which candidate method. You will test these methods and choose the best method for submission. You only need to submit code for the strengthening method you chose.** You will need to write a justification of why you chose the particular method you chose, based on your evaluation and any theoretical evidence that is appropriate.

You will implement your method such that you can generate a set of strengthened passwords from a set of dictionary words no more than 10 characters in length. 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. **Note that you can only modify existing characters, but not add characters.**

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-f16/password_cse543.tgz`.

Study the Schmidt paper [3] to learn how to evaluate your password strengthening methods in a manner that accounts for knowledge of the strengthening methods. I will apply the approach from the Schmidt paper to train my cracker to evaluate your password strengthening method.

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.txt` 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 of the 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. For example, `rockyou.txt.6.4.a` is the first of four files with at least one lower case letter, upper case letter, numeric, and symbol.

- **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 and Markov chain training data.

- **pcfgc**: This program uses the PCFG method to crack passwords.

  ```
  pcfgc 1 test-file pcfg-file > crack-file
  ```

  The `test-file` should be the password file that you generated from dictionary words and the `pcfg-file` should be PCFG and Markov chain training data produced above. Recall from the Schmidt paper that the Markov chain is used to predict characters for substrings.

  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. Asmit De, Jonas Wang, Yunqi Zhang
2. Jeffrey Acquaviva, Sravya Adavi, Kang-Lin Wang
3. Ashley Huhman, Neeraj Karamchandani, Yu-Tsung Lee, Ruochen Zhang
4. Frank Liu, Meghan Riegel, Mayank Pahadia
5. Aditya Basu, Anindita Bandyopadhyay, Asha Veerabhadraiah
6. Joseph Sharp, Sha Liu, Nicolas Papernot
7. Hanling Zhang, Morteza Ramezani, Yuquan Shan, Yuanyi Sun
8. Srikumar Sridhar, Jing Zhao, Michael Wheatman
9. Saurabh Kaul, Lidong Luo, Ryan Sheatsley
10. Chetan Sharma, Manali Latkar, Richard Heidorn
11. Daniel Krych, Robert Brotzman Smith, Chun-Yi Liu

6 Deliverables

Please submit the following:

1. Your password strengthening program, the supporting files to build the program (e.g., Makefiles), and a README with instructions on how to build and operate.

2. Output from `pwdstats` from use of `pcfgc` on the rockyou password database.

3. A writeup (no more than 5 pages) that contains three sections: (1) background on password strengthening and password cracking (based on your understanding of the research papers); (2) one candidate strengthening method per student in the group (please identify the student responsible for each) and their justifications. Given these option, provide a justification for the password strengthening method you chose - comparing to alternatives; (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. Code for password generation and the new strengthening method: I can build and run your stuff (20 pts)

2. Writeup: Clarity and foundation for technical arguments (20 pts)

3. Algorithm testing: Should improve passwords over a naive strengthening approach (20 pts)

In addition, the team with the “best” password generation algorithm, per my testing, will win a 10% bonus (6 points).

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.