CS 5435:
Security and Privacy Concepts in the Wild
Homework #3

Due: Before class on 28 Oct. 2014

100 points (+ 10 bonus)

Instructor: Ari Juels TA: Sid Telang

In this homework, you'll invent algorithms for generating **honeywords**, fake passwords that look real to an adversary.

A honeyword generation algorithm in this assignment will have the following form. It will take as input a positive integer n and a true password P consisting of a string of up to 256 ASCII characters. It will output a list $P_1, \ldots, P_n$ of n sweetwords, exactly one of which is identical with P.

Your task is to code up three algorithms that generate honeywords given a training set T, i.e., set of example passwords, for each of the following cases:

1. T is the empty set, i.e., the algorithm uses no example passwords.
2. T is the set of the 100 most common RockYou passwords.
3. T is the full RockYou dataset.

You can find the list of RockYou passwords, with frequency counts, at http://downloads.skullsecurity.org/passwords/rockyou-withcount.txt.bz2.

Your algorithms may be distinct or may be instances of a single master honeyword algorithm.

We will assign you to teams of three or four. Each team will collaboratively devise and code up the three algorithms. We ask that you use Python for this assignment.

For each of your honeyword generation algorithms, we additionally ask that you write a brief description (at most half a page) detailing your honeyword generation strategy / techniques. As usual, written descriptions of your ideas should be produced individually.

Please submit both your written description and your team's code. It is sufficient for one team member to submit the code.

You will be graded on your conceptual understanding and the quality of your ideas, primarly as reflected in your writeup. You will *not* be graded on the performance of your algorithm, i.e., how well the resulting honeywords turn out to resist attack in Homework #4.

An example honeyword generation algorithm is available at http://people.csail.mit.edu/rivest/honeywords/gen.py. You may find helpful ideas in [1, 2, 3, 4, 5].

After submission

After you've submitted your assignment, we'll provide you with a list of true passwords and ask you to generate a set of sweetwords for each of these passwords. These lists will be used in Homework #4, in which the teams of Homework #3 will try to break one another's honeyword generation algorithms. In Homework #3, you should think about the strategies you expect to use and thus expect your adversaries (classmates) to use in Homework #4...

Bonus: Honeyrides

The Citibike bicycle rental program publishes historical data on the use of its system at http://www.citibikenyc.com/system-data.

For extra credit write a program that creates "honeytrips"—false individual records of bike trips in the Citibike system. (These records might be useful, for instance, in enhancing the privacy of the true records.)

Your algorithm should take as input a monthly trip history spreadsheet, and output the same spreadsheet enhanced with 100 honeyrides.

You may use as your training set T the spreadsheets for all months *except* August 2014.

For Homework #4, we'll ask you to use your algorithm to lace the August 2014 trip history with 100 honeyrides. Teams will then try to identify honeyrides in one another's spreadsheets. You will receive a number of bonus points according to the following formula. Let h be the maximum number of honeyrides an opposing team is able to identify in your spreadsheet using their attack algorithm. You will receive $max(10 - h, 0)$ bonus points.

We rely on you (according to the CS 5435 code of gentlewomanly / gentlemanly behavior) not to download or inspect the August 2014 trip history.

References

[1] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy*, pages 538–552, 2012.

[2] A. Juels and R. Rivest. Honeywords: Making password-cracking detectable. In *ACM Conference on Computer and Communications Security – CCS 2013*, pages 145–160. ACM, 2013.

[3] P.G. Kelley, S. Komanduri, M.L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L.F. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *IEEE Symposium on Security and Privacy (SP)*, pages 523–537, 2012.

[4] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In *IEEE Symposium on Security and Privacy (SP)*, pages 162–175, 2009.

[5] Y. Zhang, F. Monrose, and M. K. Reiter. The security of modern password expiration: an algorithmic framework and empirical analysis. In *ACM CCS*, pages 176–186, 2010.