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

Due: 22 Oct. 2017
(100 points)

Instructor: Ari Juels TA: Bijeeta Pal, Fan Zhang

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 $P_1, \dots, P_n$ of n sweetwords, exactly one of which is identical with P.

Your task is to code up three algorithms (in three distinct scripts) that generate honeywords given a training set T, i.e., set of example passwords, one 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 not use any other data.

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 per algorithm, and ideally shorter) detailing your honeyword generation strategy / techniques. *Written descriptions of your ideas must be produced individually, i.e., you may collaborate on idea generation and programming, but not on your submitted written algorithm explanations.* Your written description should be sufficient for someone who isn't a member of your team to produce code that behaves very similarly to your own.

Your programs, *one for each algorithm*, should have the following command line interface:

```
> python your_program.py n input_filename output_filename
```

The file `input_filename` contains a list of real passwords, one password per line. Your program should read `input_filename` and, for each password it contains, generate a set of n sweetwords (e.g., $n = 10$). Sweetword sets should be written to a new file named `output_filename`. A sweetword set should be output as a sequence of comma separated values on a single line. Note therefore that `input_filename` and `output_filename` should have the same number of lines.

Each student should submit *both* her/his written description and her/his team's code.

You will be graded on your conceptual understanding and the quality of your ideas, primarily 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, although we may award bonus points and/or prizes for particularly good algorithms.

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 select a list of true passwords and use your algorithms 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... Think adversarially!

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.