Name: Fnu Gan Chong Yee
Class: CS 5435 – Security and Privacy Concepts in the Wild
Title: Homework 4

The algorithm described below was tested against our previous assignment with an accuracy of 21%.

The honeyword cracking approach is described in Section 1, Section 2 and Section 3 respectively.

An input file containing a list of sweetwords are passed.

The relevant split and replace functions are used to be able to access each word in a sweetword set separately, and to pass it back in the same format.

Section 1 – casing of words

Here, we consider the casing of individual alphabets in the set of honeywords.

The main criteria here involves extracting strings with all lowercase letters and those with uppercase only in the first letter:

- for each word, word.islower() and word[0].isupper() was used to check this condition.
- as some passwords contained two words (e.g. "the Boss"), an additional conditional statement (' ' in word) was used to only extract words without empty spaces in between

This was done because as per our viewing of most existing real passwords, no words have been randomly capitalized in between the word. Such capitalizations would be hard for the user to remember his/her password.

Section 2 – getting rid of appendations

Extracting common segment words (if you have apple19923, 32apple1992 and apple1992, you will return apple1992)

- All letters are first turned to lowercase using line.strip().lower()
- We are using the editdistance library to make use of Levenshtein distance.

- The Levenshtein distance between string1 and string2 is the minimum number of steps to change string 1 to string2. This means how many letters you must change, or how many letters you must add,delete from string1 so that it becomes string2.
- As the package can count this, all we need to do is compare each word in a set with each other, then only return the segment of the work which gives you the minimum Levenshtein Distance.

Section 3 – Removing Rockyou passwords

We do this because in our previous assignment, we leverage this dataset to generate honeywords.

- Calculate Levenshtein distance of Rockyou with every word in the sweet word set
- If the Levenshtein distance is lower than a threshold, then delete it from the sweetword (this means that the word is very similar to a rockyou word, and probably fake)

Lastly, we also thought of including the case in which alphabets were substituted for special characters, but we could not implement this functionality by the time of the deadline.