Brandon Plaster

CS 5435 – Homework 3

October 29th, 2014

Much of the code uses the concept of the Levenshtein distance. The Levenshtein distance between two words is the number of character manipulations required to transform one word into another word.

The code is structured into three main constructs: “pollinate”, “pollinateMe”, and “honey\_functions”.

The function “pollinate” takes in a password (p), a desired number of honey words (k), and the number of points desired to be analyzed from the rock you dataset (data\_size). It returns list of (k+1) passwords, including the original password (p).

The function “pollinateMe” takes in a seed word (p) and the set of current words in the honeypot and returns the honeypot with additional honeywords that use the seed word (p) for generation.

The set of functions “honey\_functions” contains all the functions which take a seed word (p), manipulates it, and returns a modified honeyword. Currently, there are 7 active functions. The “l33t\_word” function switches a character within a word for its leet counterpart (i.e. an ‘s’ becomes a ‘$’, and an ‘a’ becomes a ‘@’). The function ‘year\_tweaker’ finds a year within the seed word, and changes it to another year. The ‘head\_tweaker’ and ‘tail\_tweaker’ functions find numbers at the beginning and end of a string, respectively, and changes them to another number. The ‘lower\_case’ and ‘upper\_case’ functions vary the lower and upper case characters of a given string. The ‘delta\_word’ function changes numbers throughout a word.

For the case of 0 data points (i.e. no use of the RockYou data set), the function calls “pollinateMe” k-times until the honeypot is full. The way the “pollinateMe” generates a honeyword is that it chooses a random set of honey\_functions on the seed word, and if the Levenshtein distance between the honeyword and the seed word is small enough (i.e. if the two words are considered similar enough) then the honeyword is added to the honeypot. The idea behind using the Levenshtein distance is to make a set of words that are similar to the seed word, and as a result, make the entire returned set self-similar. The attempted effect is to generate a list of words that are all equally as likely.

For the case of X data points (i.e. the first X points of the RockYou data set), the function runs through the first X data points of the RockYou data set and finds the k number of words with the smallest Levenshtein distance to it. It then uses a random choice from this set and the original password to generate a list of honeywords. This set is termed as the set of seed words. In both functions, additional seed words are added by running through the initial set of seed words and adding slightly modified versions of the original seed words.