CS5435: Security and Privacy concepts in the wild

Fall2014

Homework Assignment 3 Write-Up

Honey Words

Jai Chaudhary

jc2855@cornell.edu

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 P1,...,Pn 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:

Problem 1:

T is the empty set, i.e., the algorithm uses no example passwords.

Solution

- The approach abstractly, is to have a uniform probability of a honey word being the password.
- Since, we have no example passwords. Generation using Probabilistic password model is not feasible.
- The approach is to tweak /munge the password and generate honey words that are equally likely to be chosen by a user as the input password was.
- The following heuristics were implemented as transformations to the password
    - Jumble
        - Tokenization : The idea is that since users think of characters, numbers and special characters as substrings.
        - Permutations of substrings(number, word and character) is equally likely to be generate by user
    - Substitution
        - The idea is that you can substitute special characters as they don't have different meaning to the user, Eg- !,@,%, #
        - This is valid if you replace the number sequences with random sequences. But this is not a good idea if the sequence has a pattern like 123, 1991.
        - The substitution can be at the level of strings as well but it is better with a password model.
    - Truncation
        - Since it is harder to add information. The idea is to randomly delete substrings.
        - The character-level truncation is more sensible for numbers and special character sequences.
    - Addition
        - The addition of character to a string is not a good idea.
        - The addition of very frequent substrings like 123, ! , qwerty, keep the probability almost same
    - Case-Change
        - We also considered change of case (camel, lower, upper) as it preserves the meaning

        - However due to huge bias towards lower case passwords and to avoid false alarms by users, this was rejected.
- Randomized Transformation: In order to prevent a modeling of the transformations from a large number of honeywords. All the above transformations can be applied randomly

Problem 2:

T is the set of the 100 most common RockYou passwords.

The advantage of having top 100 most common passwords is that, now

- We can instead of having a single sugarword to munge, we can have a cocktail of multiple sugarwords, DB. Those sugarwords are then chaffed(by Solution to Problem 1) to get honeywords.
- The sugarwords are actual passwords drawn from RockYou password. Their selection in this small subset is random
- We can calculate the most frequent substrings in those passwords like 123. These substrings are then appended to sugarwords to create honey-words.

Problem 3:

T is the full RockYou dataset.

Now that we have the complete RockYou password,

- The algorithm changes from Problem 2 via a change in strategy to choose sugarwords.
- Assuming the chaffing creates honeywords with the same probability as sugarwords.
- The choice of sugar words is made such that they have the same probability of being chosen as the password.
- We have an estimate of user's probability of choosing the password.
- The probability of a user choosing the input password is calculated using the nearest Rock you password as Levenshtein distance.

Bonus: Honeyrides

The characteristic of honeyrides follows from the same abstraction as above, like a reverse anomaly detection problem i.e. creating observations that seem genuine.
In the above abstraction predictor was the user and observation was the password. We were trying to come-up with observations that had the same probability as observed variable.
The honeyrides can be defined as

- the rides a user was as probable to make as the original rides he/she made.
- the user's (represented by user data) that could have made the ride with same probability as original user.