Second Order Markov Model Based Proactive Password Checker

Harpreet Singh Dhillon
Department of Electronics and Communication Engineering, IIT Guwahati, India.
Roll No.: 04010214; email: harpreet@iitg.ernet.in

Abstract— **With the rapid increase in the multiuser systems, the issues relating to the password security have become very important. The problems inherent in allowing users to choose passwords without restriction have been widely reported in literature. Proactive password checking has been a common means to enforce password policies and prevent users from choosing easily guessable passwords in the first place. One such proactive password checking scheme, based on second order Markov model, is discussed in this report.**

Index Terms— **Proactive password checking, dictionary attack, markov model, password security.**

I. Introduction

The problems inherent in allowing users to choose passwords without restriction have been widely discussed in literature. Due to the limitation of human memory, people are inclined to choose easily guessable passwords (e.g. phone numbers, birthdays, names of family or friends, or words in human languages) that might lead to severe security problems. Though it was commonly believed that secure passwords were difficult to remember and easy-to-remember passwords were insecure, a recent experiment [1] showed with hard data that passwords based on mnemonic phrases could provide both good memorability and security, but non-compliance with password selection advices was a main threat to password security.

Proactive password checking [2] has been a common means to enforce password policies and prevent users from choosing easily guessable passwords in the first place. When a user chooses a password, a proactive checker will determine whether his password choice is acceptable or not, and this proactive checking is done online and the user will be immediately responded the result. Among common approaches to improving password security by selecting good passwords, such as user education, program-controlled password generation and reactive password checking (i.e., system administrators periodically run password cracking programs to search weak passwords), proactive password checking has been widely regarded as the best.

This report is submitted as a part of MA–402 (Queueing Models for Performance Analysis) course taken by Dr. N. Selvaraju, Asst. Professor, Department of Mathematics, IIT Guwahati, India.

II. Proactive Password Checking

In proactive password checking scheme, a user is allowed to select his own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

The trick with the proactive password checker is to strike a balance between user acceptability and password strength. If the system rejects too many passwords, users will complain that it is too hard to select a password. If the system uses some simple algorithm to define what is acceptable, this provides guidance to the password crackers to refine their guessing techniques.

Simple proactive techniques involve some kind of rule enforcements. For example, all passwords must be at least eight characters long or in the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits and punctuation marks. Although this approach is superior to simply educating the users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as to which passwords not to try but may still make it possible to do password cracking.

Another possible procedure is simply to compile a large dictionary of possible “bad” passwords. When the user selects a password, the system checks to make sure that it is not on the disapproved list. There are two main problems of space and time with this approach. For dictionary to be effective, it has to be very large, hence leading to a large memory requirement. The time required to search a large dictionary may itself be very large. Researchers have been looking for good

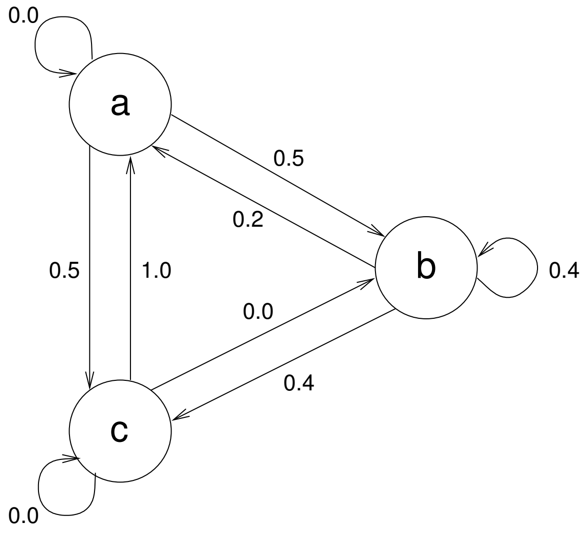

Fig. 1. Transition probability diagram for the simplified first order markov model.

algorithms that could achieve both fast checking speed and effective dictionary compression at the same time. One such password checker, based on Markov model is discussed in the following section.

III. BApasswd: A Proactive Password Checker

This proactive password checker, named as BApasswd [3]–[4], is based on the Markov model for the generation of the guessable passwords. Simplified illustration of this model is given in Fig. 1. This model shows a language consisting of an alphabet of three characters. The state of the system at any time gives the identity of the most recent letter. The value on the transition from one state to another represents the probability that one letter follows another. Thus, the probability that the next letter is a, given current letter is b is 0.2.

In general, a Markov model is a quadruple $[m, A, T, k]$, where m is the number of states in the model, A is the state space, T is the transition probabilty matrix and k is the order of the model. For a k^{th} order model, the probability of making a transition to a particular letter depends on the previous k letters that have been generated. Fig. 1 shows such a simple first order model. The transition probability matrix for this model can be written as follows.

$$\mathbf{T} = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

The authors of [3] report on the development and use of the second-order model. To begin, a dictionary of the guessable passwords is constructed. Then the transition matrix is calculated as follows:

1) Determine the frequency matrix f, where $f(i, j, k)$ is the number of occurances of the trigram consisting of the i^{th}, j^{th} and k^{th} character. For example, the password $happy$ yields the trigrams hap, app and ppy.
2) For each bigram ij, calculate $f(i, j, \infty)$ as the total number of bigrams begining with ij. For example, $f(a, b, \infty)$ would be total number of trigrams of the form aba, abb, abc and so on.
3) Compute the enteries of $\mathbf{T}$ as follows:

$$\mathbf{T}(i, j, k) = f(i, j, k)/f(i, j, \infty)$$

The result is a model that reflects the structure of the words in the dictionary. For a given password, the transition probabilites of all its trigrams can be looked up. Some standard tests can then be used to determine if the password is likely or unlikely for that model. Passwords that are likely to be generated by the model are rejected. With this model, the question “Is this a bad password?”, is being transformed into “Was this string (password) generated by this Markov model?”.

IV. Conclusion

A proactive password checker, based on second order Markel model, is discussed in this report. This model is seen to transform the problem of deciding whether the passwords is bad or good to the problem of observing whether this password was generated by the Markov model or not.

References

[1] J. Yan, A. Blackwell, R. Anderson and A. Grant, “The Memorability and Security of Passwords – Some Empirical Results”, *Tech. Report No. 500, Comp. Lab., Univ. of Cambridge*, 2000. http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/tr500.pdf

[2] F Bergadano et al., “Proactive Password Checking With Decision Trees”, *ACM conference on computer and communications security*, 1997, Zurich.

[3] C. Davies and R. Ganesan, “BApasswd: A New Proactive Password Checker”, in *proc. 16th National Comp. Security Conf.*, pp. 1–15, Baltimore, MD, Sept. 1993.

[4] W. Stallings, *Cryptography and Network Security*. pp. 486–488, Englewood Cliffs, NJ: Prentice-Hall, 1998.