Password Entropy and Password Quality

[redacted]

Abstract **Passwords are the first line of defense for many computerized systems. The quality of these passwords decides the security strength of these systems. Many studies advocate using password entropy as an indicator for password quality where lower entropy suggests a weaker or less secure password. However, a closer examination of this literature shows that password entropy is very loosely defined. In this paper, we first discuss the calculation of password entropy and explain why it is an inadequate indicator of password quality. We then establish a password quality assessment scheme: password quality indicator (PQI). The PQI of a password is a pair $\lambda = (D, L)$, where D is the Levenshtein's editing distance of the password in relation to a dictionary of words and common mnemonics, and L is the effective password length. Finally, we propose to use PQI to prescribe the characteristics of good quality passwords.**

I. Introduction

Authentication Authentication and authorization are the foundation of information security. Authentication is responsible for verifying that a person is really who he/she claims, and authorization is about assigning appropriate privileges to the person after the verification of his/her identity. There are 3 types of authentications [10, p 209]: (i) something the user knows, for example, password and PIN (personal identity number), (ii) something the user has, for example, physical keys, access cards, and smart cards etc., and (iii) something the user is so called biometric authentication, such as voice recognition [14], fingerprints matching, and iris scanning etc. Password authentication is simple, accurate, and effective and will continue to be the working horse of information security. According to [11], who reported the panel discussion at RSA 2005 conference, "*Password will be with us forever*", because "*We've got to make security simpler to use if it's going to be effective*", as suggested by the panel members.

The strength of password authentication relies on the strength of the passwords. Therefore, measuring the quality of password becomes an interesting topic. For example, how can we accurately measure the quality difference of passwords "akjuwfg" and "D$f9" and therefore provide a quantitative measurement on which one is better? Password entropy is mentioned as a quality indicator for passwords in many occasions. Higher entropy means better quality. Most of the literature just briefly mentions that one should choose the passwords with higher entropy, for example [15]. There are just a few which actually provide the details on the calculation or estimation of password entropy [2]. After carefully examining the literature, we realize that the concept of password entropy is actually loosely defined. The suggested estimations are far from indicating the quality of a password. The concept of information entropy was introduced by Shannon [12] to measure information content. It has been widely used in communication, coding, and cryptography etc. areas [8]. The calculation of the entropy is based on a statistic distribution model of a language and is conducted on a model of n order Markov process. This is actually the fundamental reason why entropy cannot be used as a quality indicator for passwords

- As all agreed, there is no statistic distribution for passwords. We doubt if it is possible to establish this distribution, even with a collection of large number of sample passwords. Unlike English words, where we follow certain rules to communicate with each other, choosing a password is not for communication purpose. There is no uniform rule for everybody to follow. It may therefore not exist a converge model.
- Password guessing is not a Markov process. There is no such thing as knowing the first several characters of a password and then proceeding to the following characters, where the knowledge of these leading characters can help to guess the next characters. Guessing a password is an all or nothing game. Either you get the password exactly as it is, or nothing at all. Therefore, it is meaningless to calculate password entropy based on the composing characters.
- Guessing entropy [9] is not an appropriate indicator for the quality of individual password either. It only establishes the low boundary for how many guesses needed to crack passwords.

In this paper, we advocate a different means of measuring password quality password quality indicator (PQI) [7]. We

Manuscript received May 10, 2010.

Wanli Ma is with the Faculty of Information Sciences and Engineering, University of Canberra, Australia (phone: +61-2-62012838; fax: +61.2.62015231; e-mail: Wanli.Ma@canberra.edu.au).

John Campbell is with the Faculty of Information Sciences and Engineering, University of Canberra, Australia (e-mail: John.Campbell@Canberra.edu.au).

Dat Tran is with the Faculty of Information Sciences and Engineering, University of Canberra, Australia (e-mail: Dat.Tran@canberra.edu.au).

Dale Kleeman is with the Faculty of Information Sciences and Engineering, University of Canberra, Australia (e-mail: Dale.Kleeman@canberra.edu.au).

978-0-7695-4159-4/10 $26.00 © 2010 IEEE
DOI 10.1109/NSS.2010.18

IEEE computer society

believe that the quality of a password is decided by the time required to crack the password. The longer time it requires, the stronger the password is. The PQI of a password is a pair $\lambda = (D, L)$, where D is the Levenshtein's editing distance (Levenshtein, 1965) of the password to the base dictionary words, and L is the effective password length. The effective password length is the equivalent length of the password in the standard password format, which consists of only the 10 digit characters (0-9). From PQI, we further develop a concise rule for choosing a good password: *a good password should be at least 8 characters long, with at least 3 special characters plus other alphanumeric characters*. The rule is easy to remember and easy to be checked by a computer program. It avoids the costly operation of proactive password checking [5,1,15].

The paper is organized as follows. We first study password entropy and then develop the rationale of our password quality indicator theory. We conclude the paper with a summary.

II. Calculating Password Entropy

Let's envisage a device which randomly generate lower case English letters (a to z), one character after another. We guess what letter is before it comes. If the device does not have any predefined patterns to follow, each of the 26 letters has the equal chance ($\frac{1}{26}$) to be generated. We just randomly guess a character out of the 26. As we cannot precisely predicate the letter to be generated, there is therefore uncertainty. The uncertainty is decided by how rare or common an event (a letter being generated) happens. In this case, it is the same ($\frac{1}{26}$) for every letter. However, if we know that the device only generates 3 letter long English words, the guess game will be different. At very beginning, without any existing knowledge about the word to be generated, we may try a more likely leading letter, say "t". After we see the second letter, say "h", we can almost certain that the third letter will be "e", and the word is "the".

Let $\Omega = \omega_1 \omega_2 ... \omega_N$ be the stream of characters (letters) generated by the device, where ω_1, ω_2, …, or ω_n takes the value from a character set $C = \{\alpha_1, \alpha_2, ..., \alpha_M\}$. In the case of the previously discussed letter generating device, C has the 26 lower case English letters. α_1 represents the letter a, α_2 represents the letter b, and so on, and $M = 26$. The possibility of the i^{th} character generated is $P(\omega_i)$, or simply P_i. The uncertainty can be calculated by the following formula:

$$u_i = \log \frac{1}{P_i} = -\log(P(\omega_i))$$

The overall uncertainty on Ω is:

$$H_0(\Omega) = -\sum_{i=1}^{N} \log(P(\omega_i))$$

This formula just simply sums each individual uncertainty together. In the calculation, we assume that the characters generated are independent from each other. The knowledge of previously generated characters does not have any impact on our guess of the next character. After seeing large amount of the characters generated by the device, we can calculate the average uncertainty of the character set:

$$H_0 = -\sum_{i=1}^{M} P(\alpha_i) \log(P_i(\alpha_i))$$

or simply

$$H_0 = -\sum_{i=1}^{M} P_i \log(P_i)$$

This is the famous Shannon entropy formula. Uncertainty is just the other name of entropy. The subscript *0* of H_0 indicates that the combinations of characters are independent from each other.

In the guessing game, if we know that the device only generates 3 characters long English words, we then can have the knowledge of already being generated characters to help us more accurately guess the next character, because the yet to come character, to certain degree, depends on the previous 1, 2, 3, or more characters. This process is thus a Markov process.

III. Markov processes

Let $X = \{X_1, X_2, ..., X_T\}$ be a sequence of T random variables, $T \geq 0$, $\mathbf{V} = \{V_1, V_2, ..., V_M\}$ be the set of M states in a Markov process. $\mathbf{V}$ is actually the character set C discussed before. There is a 1 to 1 mapping between V_{i} and α_{i}, $1 \leq i \leq M$. Consider the conditional probabilities:

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, ..., X_1 = x_1)$$

where *t* = 1, 2, …, *T* are values taken by the corresponding variables X_t, x_1, x_2, ,,, x_t take values from $\mathbf{V}$. If the event at time *t* depends only on the immediately preceding event at time *t* 1, i.e.,

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, ..., X_1 = x_1) = P(X_t = x_t \mid X_{t-1} = x_{t-1})$$

The process is called a **first order Markov process**. If we look at the guessing game discussed previously, when we see character "t", we will guess the next character `based on the knowledge of "t", and most likely, we will guess the character "h", as there is more chance for "h" to follow "t" than any other characters. However, after we see the character "h", we won't have the knowledge of "t" any more, and we only base on "h" to guess the next character. We may not be able to guess "e". To be able to guess the next character based on the knowledge of previous 2 characters, we need **second order Markov process**:

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, ..., X_1 = x_1) =$$
$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2})$$

IV. Calculating Entropy on Markov processes

If every character is independent from the others, we have **zero order Markov process**. The calculation of entropy is straightforward:.

$$H_0(\Omega) = -\sum_{i=1}^{N} \log(P(\omega_i)) \quad \text{and} \quad H_0 = -\sum_{i=1}^{M} P_i \log(P_i)$$

as discussed before. For the first order Markov process, where a character depends on its immediate previous character, we have $H_1(\Omega)$ and H_1 :

$$H_1(\Omega) = -(\log(P(\omega_1)) + \sum_{i=2}^{N} P(\omega_{i-1}) \times \sum_{j=1}^{M} P(\omega_j | \omega_{i-1}) \log(P(\omega_j | \omega_{i-1})))$$

and

$$H_1 = -\sum_{i=1}^{M} P_i \sum_{j=1}^{M} P_{ij} \log(P_{ij})$$

where P_{ij} is short for $P(X_t = x_j \mid X_{t-1} = x_i)$. For the second order Markov process, where the character depends on its immediate previous 2 characters, we have H_2 ($H_2(\Omega)$ is omitted):

$$H_2 = -\sum_{i=1}^{M} P_i \sum_{j=1}^{M} P_{ij} \sum_{k=1}^{M} P_{ijk} \log(P_{ijk})$$

where P_{ijk} is short for $P(X_t = x_k \mid X_{t-1} = x_j, X_{t-2} = x_i)$.

Let's take the second order Markov process for example. If we restrict our discussion within the words of 3 characters, the entropy of "the" is for sure lower than a randomly chosen words, say "thv", "txy", or "jzv". The calculation of the entropy is based on our understanding of the probability distributions of the letters under the context of legitimate English words. We can almost certain that the 3rd character is "e" after seeing "t" and "h".

The entropy of a password heavily depends on if it is on the list of patters or not. If it is, the entropy will be low; otherwise, high. Therefore, the problem now is reduced to: if a spelling (pattern) is on the list. The next logic question is thus what a pattern is. For example, "zoe" is a popular name. It is on the list of any password cracking dictionary, but it is not on the Fedora Core 5 Linux dictionary. By common sense, we would all agree to include "zoe" in the dictionary, and thus the entropy for "zoe" will be low, and therefore it is not a good password. However, if we go a step further, do we accept "abc", "xyz", "qwe", "qru", "qaz", and "esz" etc. as legitimate patterns? On the keyboard, you can easily realize the position patterns of the later 4 spellings. More complicated patterns can also be made in this manner. We may be reluctant to accept that these spelling are legitimate patterns, perhaps just because they are not popular or oblivious. However, when talking about passwords, we cannot expect popular or oblivious patterns. To keep all kinds of unthinkable patterns in the dictionary not only enlarges the dictionary but also impossible. Therefore, from password guessing point of view, exhaustive search is not avoidable. Performing exhaustive search is the same as assuming that every character has the exactly same probability and leads to higher entropy.

V. Entropy of guessing a password

Guessing a password is different from guessing an English word. It is not a Markov process, where we are able to more accurately guess the next character based on the knowledge of previous characters. To guess a password, we have to guess the spelling completely right. There is no partially correct guess   some characters are right, and the others are wrong. Guessing entropy was proposed by Massey [9] to estimate the average number of successive guesses of a set of passwords. The lower boundary is:

$$E(G) \geq \tfrac{1}{4} \times 2^H + 1$$

Guessing entropy is about the low boundary for the number of successive guesses. To the contrary of some claims, it is inappropriate to be used as the measurement of the quality of individual password.

Due to the all or nothing nature of guessing a password, it is meaningless to calculate the entropy of a password based on its composing characters. Without the knowledge of the characteristics of the password, the only assumption we can make, apart from trying our luck with a big dictionary, is that every character has the same probability. This assumption leads to maximum entropy. The passwords of the same length have the same amount of entropy. As the length becomes longer, the entropy of the password goes up, and the average entropy per character stays the same.

In summary, entropy is not a suitable measurement for the quality of individual passwords.

VI. Password Quality Indicator

There are many different types of password attacks [3,4]. In essence, password attacking is about trying different character combinations until getting a match to the right password. To effectively crack a password, some strategies have to be in place. The obvious combinations should be tried before the brute force enumeration of all possible password candidates. In general, a likely path to crack a password is, in the order of:

1. trying dictionary words,

2. trying 1 (and perhaps 2) character variations to the dictionary words,
3. trying to enumerate all possible spellings of a smaller character set, say just lower case characters or all lower case characters plus digit characters, and finally,
4. brute force enumeration of all possible password candidates with the full character set (93 characters).

The quality of a password depends on how long it takes to find out the right match. The longer it takes, the better the quality is. Thus, we can measure the quality of a password by *how different it is from the dictionary words*, *how long it is*, and *how big the password character set is*.

Levenshtein's editing distance [6] can accurately measure how different two strings are. This metric calculates the distance between two strings by counting the minimal number of single character manipulations required, such as an insertion, deletion, or modification, to make the 2 strings the same [13]. For example, the distance between “zoe” and “coe” is 1, and “the” and “zoe” 2. To measure how different a password is from all the base dictionary words, first, we line up all the dictionary words, and then, we check the Levenshtein's editing distance of the password against every single word on the line. The minimum distance is the distance of the password to the base dictionary words.

The length of a password is the number of characters in the password. It plays a vital role in deciding how long it takes to crack the password.

A password is made of characters, which are from certain groups, e.g., all characters, low case alphabet characters, or digit characters. We call these groups character sets. We artificially group the 93 printable characters into 4 sets:

- Character Set 1: 26 lower case letters: abcdefghijklmnopqrstuvwxyz
- Character Set 2: 26 upper case letters: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Character Set 3: 10 digit characters: 01234567890
- Character Set 4: 31 special characters: ~!@#$ %^&*( )_-+= {}|[] \:”<> ?;’,. /

To measure the character sets used in a password, we propose password complexity index (PCI). We assign PCI value 26 to Character Set 1, 26 to Character Set 2, 10 to Character Set 3, and 31 to Character Set 4. If a password contains a character from Character Set 1, the value 26 is added to the PCI of the password, so long and so forth. However, the value of each Character Set is only used once, i.e., the second and the subsequent character in the same Character Set do not add any extra value to the password PCI. If the characters of a password only draw from Character Set 3, i.e., only 10 digit characters (0-9), the password has the *Standard Password Format*. A password in the standard password format has PCI 10. For example, passwords “125467” and “98456902” are in the standard password format, but “s125467”, “8765t”, and “ast+Ugh” are not.

For a password Ω, which has PCI value *c* and length *m,* the number of all possible password candidates of the same format is c^m. To have the same number of password candidates in standard password format, which has PCI value of 10, we need to find out the length (*L*) of the password candidates in standard password format. Thus, we have $c^m = 10^L$. Therefore, $L = m \times \log_{10} c$. We call *L* the *effective length* of password **P**. For example, the effective password lengths of passwords “akjuwfg” and “D$f9” are 7.98 and 7.88 respectively.

The *password quality indicator (PQI)* of a password is a pair $\lambda = (D, L)$, where *D* is the Levenshtein's editing distance of the password to the base dictionary words, and *L* is the effective password length. When *D>=3* and *L>=14*, we have a good password. *D>=3* means that the password is at least *3* characters different from the base dictionary words, and *L>=14* means that there are at least 10^{14} possible password candidates to be tried to crack the password.

The easiest way to achieve *D>=3* is to have *3* special characters (from Character Set 4) in the password. The requirement is easy to remember and also easy to implement with programs to check if a password meets the requirement. Of cause, by using *3* special characters to make *D>=3*, we do miss a number of password candidates which are *3* Levenshtein's editing distance units away from the base dictionary words. However, this simplified solution is justified for several reasons. First, *3* special characters guarantee the password has at least 3 units of Levenshtein's editing distance to the base dictionary words. Second, it is easy to remember for the end users. Third, it is also easy to implement by computer programs to perform proactive password checking. Finally, by forcing *3* special characters in the password guarantees an increase in the PCI value, and therefore, the effective length of the password.

VII. Conclusion

In this paper, we first studied password entropy and realized that it cannot be used as an indicator for password quality. Based on the assumption that the quality of a password is in proportion to the time required to crack it, we introduced password quality indicator (PQI). From PQI, we further develop a simple rule for choosing and checking a good quality password *at least 8 characters long, with at least 3 special characters plus other alphanumeric characters*.

References

[1] BLUNDO, C., D'ARCO, P., SANTIS, A. D. & GALDI, C. (2002) A novel approach to proactive password checking. IN DAVIDA, G. I., FRANKEL, Y. & REES, O. (Eds.) International Conference on Infrastructure Security (InfraSec 2002). Bristol, UK, Springer.

[2] BURR, W. E., DODSON, D. F. & POLK, W. T. (2006) Electronic Authentication Guideline: Recommendations of the National Institute

of Standards and Technology. 1.0.2 ed., National Institute of Standards and Technology, USA. http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[3] CAMPBELL, J., KLEEMAN, D. & MA, W. (2006) Password Composition Policy: Does Enforcement Lead to Better Password Choices? 17th Australasian Conference on Information Systems (ACIS 2006). Adelaide, Australia.

[4] CAMPBELL, J., KLEEMAN, D. & MA, W. (2007) The Good and Not So Good of Enforcing Password Composition Rules. Information Systems Security, 16, 2-8.

[5] CISNEROS, R., BLISS, D. & GARCIA, M. (2006) Password auditing applications. Journal of Computing in Colleges, 21, 196-202.

[6] LEVENSHTEIN, V. (1965) Binary codes capable of correcting deletions, insertions, and reversals. Problems in Information Transmission, 1, 8-17.

[7] MA, W., CAMPBELL, J., TRAN, D. & KLEEMAN, D. (2007) A Conceptual Framework for Assessing Password Quality. International Journal of Computer Science and Network Security, 7, 179-185.

[8] MACKAY, D. L. C. (2005) Information Theory, Inference, and Learning Algorithms, Cambridge University Press (also accessible from http://www.inference.phy.cam.ac.uk/mackay/itila/).

[9] MASSEY, J. L. (1994) Guessing and Entropy. IEEE International Symposium on Information Theory.

[10] PFLEEGER, C. P. & PFLEEGER, S. L. (2003) Security in Computing, Prentice Hall.

[11] SAITA, A. (2005) RSA 2005: Passwords at the breaking point.

[12] SHANNON, C. E. (1948) A Mathematical Theory of Communication. Bell System Technical Journal, 27, 379-423, 623-656.

[13] STEPHEN, G. (1994) String Searching Algorithms, World Scientific Publishing.

[14] TRAN, D., WAGNER, M., LAU, Y. W. & GEN, M. (2004) Fuzzy Methods for Voice-Based Person Authentication. IEEJ (Institute of Electrical Engineers of Japan) Transactions on Electronics, Information and Systems, 124, 1958-1963.

[15] YAN, J. (2001) A Note on Proactive Password Checking. ACM New Security Paradigms Workshop. New Mexico, USA.