Password Verification Against Online Guessing Attack

Albert Guan (官振傑)

2024/10/14

Contents

1. Introduction
2. Preliminaries
3. Proposed Password Verification Scheme
4. Analysis of the Proposed Scheme
5. Parameter Selections
6. Simulation of the Proposed Scheme
7. Conclusions

Introduction

Username and password are the most common method for authenticating users on servers and social networks accounts.

Let u be a user, and p_u be the password chosen by u, and

$$w_1, w_2, \ldots, w_t (= p_u)$$

be the passwords entered by a user in a login section.

In the current password authentication system, once the password is correct, the user's access right is granted.

Can we distinguish between a user and an attacker trying to guess a password?

Strong passwords

1. Contains different categories of characters
   - A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
   - a b c d e f g h i j k l m n o p q r s t u v w x y z
   - 0 1 2 3 4 5 6 7 8 9
   - + - * / ···
2. Length at least 8
3. Do not use personal ID as part of user passwords.
4. Do not use personal information, such as birthday, as part of user passwords.

Password Issues

- Strong passwords are usually difficult to guess, but also difficult for users to remember.
- Many users use passwords that are easy to remember. These passwords are also easy for attackers to guess.
- Password guessing attacks are a serious security threat to servers and social networks accounts, but no effective countermeasures were proposed.

We show that comparing entropy of the passwords input by a user is an effective and efficient way to distinguish between a legal user and an attacker, even if a user selects a common password as his password.

Related Works

1. (offline)
   S. Houshmand, S. Aggarwal, and R. Flood, Next generation PCFG password cracking. *IEEE Transactions on Information Forensics and Security*, 10(8):1776–1791, 2015.

2. (online)
   Y. Tian, C. Herley, and S. Schechter. Stopguessing: Using guessed passwords to thwart online password guessing. *IEEE Security & Privacy*, 18(3):38–47, 2020.

Online password guessing attacks are still considered difficult to counter.

Passwords Verification – An Example

user password = welcome, $t = 5$

	passwords input by user					entropy
A	admin	Welcome	WelCome	welcome		$(H_0(A) = 3.700)$
B	123456	qwerty	admin	abc123	welcome	$(H_0(B) = 4.459)$

Which one is more likely to be a legitimate user, A or B?

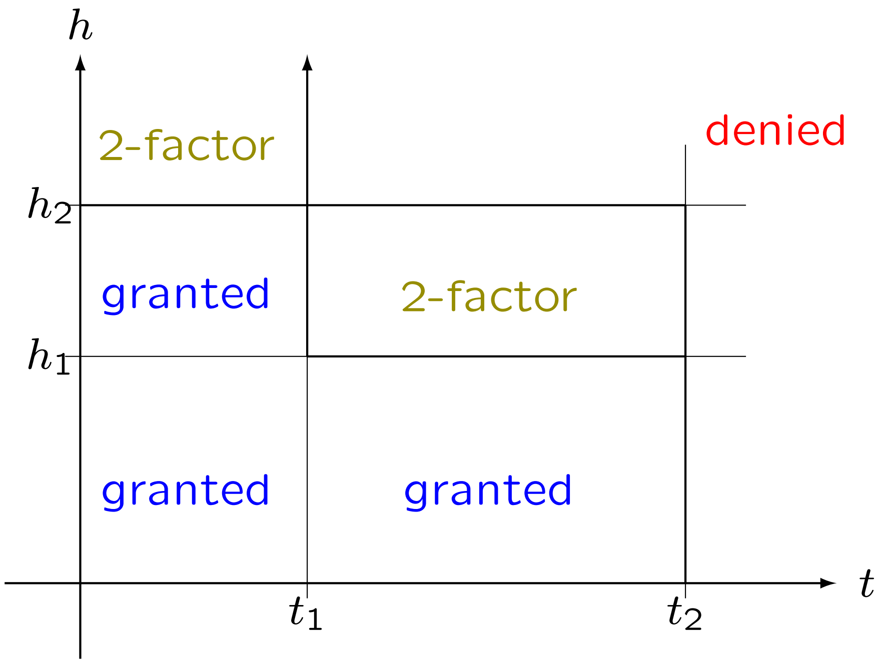
Basic Assumptions

- A legitimate user knows his password, but
- An attacker can only guess what the password is.

Our new password verification scheme

- computes the entropy $h(w)$ of $w = w_1 w_2 \cdots w_t$,
- uses t and $h(w)$ to decide whether the user is legitimate or not.

Password Verification Scheme



Main Features of Our Password Verification Scheme

1. Effectiveness:
   It can easily distinguish legitimate users from attackers even when the password is common password.

2. Computationally lightweight:
   Computing maximum entropy $H_0(X)$ only need to record number of distinct characters in $w = w_1, w_2, \ldots, w_t$.

3. Keep all passwords confidential:
   Our scheme does not compute edit distances $d(w_i, p_u)$, nor does it need to store commonly used passwords.

4. Can be integrated into existing scheme easily:
   Our novel password verification scheme is a slight modification of the commonly used scheme.

Rényi entropy

The Rényi entropy of order α is defined as

$$\mathsf{H}_\alpha(X) = \frac{1}{1-\alpha}\log\left(\sum_{i=1}^{n} p_i^\alpha\right).$$

- $H_0 \geq H_1 \geq H_2 \cdots \geq H_\infty$.
- If $p_i = 1/n$, then $H_\alpha = \log(n)$.
- H_0: Hartley (maximum) entropy: $\mathsf{H}_0 = \log(m)$, where m is the number of non-zero p_i's.
- H_1: Shannon entropy: If $\alpha \to 1$, then $\mathsf{H}_1 = -\sum_{i=1}^{n} p(i)\log p(i)$.
- H_∞: Minimum entropy: $\mathsf{H}_\infty(X) = -\log(\max_i\{p_i\})$.

Simulation Results

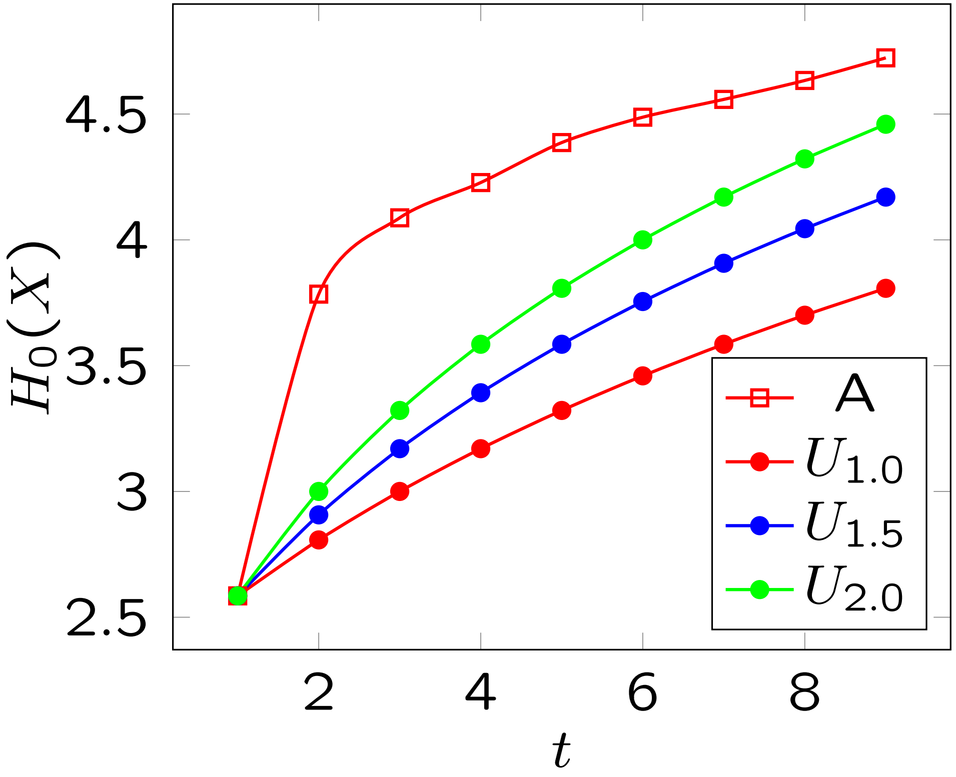
The simulation is based on:

- A user makes only a few typing mistakes for each password.
- An attacker randomly chooses a password from a list of common passwords from 2011 to 2019, compiled by SplashData.

		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
H_0	U	2.58496	2.80735	3.00000	3.16993	3.32193
	A	2.58496	3.78856	4.06747	4.24400	4.39096
H_1	U	2.52164	2.79359	2.93996	3.03671	3.10784
	A	2.52164	3.60544	3.86130	4.02327	4.14324
H_2	U	2.44478	2.40526	2.40960	2.41504	2.41936
	A	2.44478	3.40556	3.64259	3.80062	3.90026
H_∞	U	1.80735	1.80735	1.80735	1.80735	1.80735
	A	1.80735	2.61103	2.77526	2.91697	2.96229

Experimental results

Assume that the user makes $x = 1.0, 1.5, 2.0$ typing mistakes in average in each login attempts.



General Model of User and Attacker Behavior

- Let $\mathcal{P}$ be the set of all possible passwords that a user or an attacker can enter in a login session.

- The set $\mathcal{P}$ is usually finite, and it should include the password the user chooses for his/her account.

- Let $\mathcal{P} = \{P_1, P_2, \cdots P_N\}$, and p_i be the probability of choosing a password $P_i \in \mathcal{P}$ to enter into the password verification system.

- The behavior of a user or an attacker is to repeatedly select a password P_i from $\mathcal{P}$, with the probability p_i, until the correct password is entered or the number of allowed login attempts is reached.

Behavior of a User

- The set of all possible passwords for a user is referred to as the dictionary of the user, denoted by $\mathcal{P}_u$.
- Because the user knows the password of his/her choice, but may make mistakes when entering the password, the set $\mathcal{P}_u$ contains the password P, and strings with a small edit distance from P.
- The probability of each $P_i \in \mathcal{P}$ may be uniform, or inverse propositional to the edit distance $d(P, P_i)$.

Behavior of an Attacker

- The set of all possible passwords for an attacker is referred to as the dictionary of the attacker, denoted by $\mathcal{P}_a$.
- For an attacker who does not know the user password P, he/she can only guess what P is.
- The attacker is likely to use a set of commonly used passwords as the dictionary to generate guesses, rather than randomly guessing passwords from the universal set.

Most password cracking tools, including John the Ripper, also rely on this dictionary of commonly passwords to speed up guessing attacks.

Therefore, it is reasonable to assume that the set $\mathcal{P}_a$ is the set of commonly used passwords collected from past research.

Attacker's Best Strategy

Assume that the attacker does not have additional information about the user's password P, except that $P \in \mathcal{P}$.

According to probability theory, the best strategy for the attacker is randomly and uniformly chooses a password from his/her dictionary $\mathcal{P}$, that is, $p_i = 1/|\mathcal{P}|$.

More on Attacker's Tactics

- In some situations, the attacker can make use of side information to guess the correct password.

- In the case that the attacker knows some information about the user's password, he may have a better strategy to correctly guess the password P.

- This can be modeled by changing the probability p_i of each password $P_i \in \mathcal{P}$.

Formal Problem Formulation

The online password guessing attack detection problem can be formulated as:

> Given a set of t passwords $\{P_1, P_2, \cdots, P_t\}$, determine whether these passwords come from the user's dictionary $\mathcal{P}_u$ or the attacker's dictionary $\mathcal{P}_a$.

Password Entropy Estimations

Our study shows that computing the entropy of the concatenation of all the passwords entered by a user in a login session $P_1 P_2 \cdots P_t$ can solve the problem efficiently.

In theory, by choosing a correct dictionary of passwords $\mathcal{P}_u$ (P_a) and the probability of each password in $\mathcal{P}_u$ ($\mathcal{P}_a$), we can estimate the entropy of the passwords entered by the user (attacker) in a login session.

In practice, it may be difficult to determine the set $\mathcal{P}_u$ ($\mathcal{P}_a$) and the probability of each password in $\mathcal{P}_u$ ($\mathcal{P}_a$).

Password Entropy Estimation

We develop an approximate method to estimate the entropy of the user and the entropy of the attacker.

Without loss of generality, assume that Rényi entropy of type 0 $H_0(X)$ is used in the computation of the entropy of the passwords.

Password Entropy Estimation

First, we define the notations used in the approximation method.

- Let P and Q be two strings over the same alphabet.
- Let $h(P, Q)$ be the number of distinct characters c such that c occurs in Q but not in P.
  Note that $h(P, Q)$ is not symmetric, i.e. $h(P, Q)$ may not be equal to $h(Q, P)$.
- Let ϵ denote the empty string.
  Then $h(\epsilon, P)$ is the number of distinct characters in P, and the entropy of P is $\log(h(\epsilon, P))$ if Rényi entropy of type 0 is used.

Password Entropy Estimation

- Let $P_1, P_2, \cdots, P_t$ be the sequence of the passwords entered by a user or by an attacker.
- Let Q_i be the concatenation of $P_1 P_2 \cdots P_i$, for $i > 0$.
- It is clear that each password P_i may increase the entropy of the password string defined by Q_i, which is the concatenation of $P_1 P_2 \cdots P_i$.
- Let $\delta_i = h(Q_{i-1}, P_i)$ be the number of new characters induced by the password P_i.

Password Entropy Estimation – δ_i

- One cannot expect the value of each δ_i to be the same, or almost the same, for every $i = 2, 3, \cdots, t$.
- Even if for a legitimate user, he/she may enter a password for another account, and the two passwords can be quite different. For example: $P_1 =$ Admin, $P_2 =$ welCome, $P_3 =$ Welcome.
- On the other hand, if P_i is a permutation of the characters in the previous entered password, then $h(Q_{i-1}, P_i) = 0$. Therefore, the value of δ_i may vary greatly.

However, we may estimate the average value of δ_i, which is also the same as estimating the sum of all δ_i's.

Parameter Selection

The behavior of a legitimate user and an attacker can reasonably be modeled by:

$$\sum_{i=1}^{t} \delta_i \leq t \cdot \alpha \cdot h(\epsilon, P_t),$$

where α is a constant.

The sum $\sum_{i=1}^{t} \delta_i$ is roughly equal to the number of distinct characters of the correct password $h(\epsilon, P_t)$ plus the total number of errors made by the user in the login session.

Parameter Selection

Let $\alpha = \alpha_u$ be the constant for a legitimate user, and $\alpha = \alpha_a$ be the constant for a attacker. Thus, the estimated entropy of a legitimate user is

$$\log(h(\epsilon, P_t)) + \log(t\alpha_u),$$

and the estimated entropy of an attacker is

$$\log(h(\epsilon, P_t)) + \log(t\alpha_a).$$

Parameter Selection

Based on the above analysis, the parameters t_1, t_2, h_1, h_2 used in the proposed password verification scheme can be selected as follows.

- First select t_1 and t_2. Usually $t_1 = 3$, and $t_2 = 5$ to 8, depending on the security requirement of the system.

- Based on the values of t_1 and t_2, we can then estimate the entropy of a user and the entropy of a attacker when they try to login. The values of h_1 and h_2 can then be selected as the midpoint of user entropy h_1 and attacker entropy h_2, where

$$h_1 = (h(\epsilon, P_t) + \log(t\alpha_u) + h(\epsilon, P_t) + \log(t\alpha_a))/2, \text{ at } t = t_1;$$

  and

$$h_2 = (h(\epsilon, P_t) + \log(t\alpha_u) + h(\epsilon, P_t) + \log(t\alpha_a))/2, \text{ at } t = t_2.$$

Conclusions

1. We propose an effective approach to address online password guessing attacks.

2. The proposed password verification method can be implemented efficiently.

3. It can be easily integrated into current password verification methods.

Password Verification Against Online Guessing Attack

以上防止線上密碼猜測攻擊之論文已發表於

IEEE Transactions on Dependable and Secure Computing

1. 資訊安全領域最重要的頂級期刊之一

2. 根據 JCR 的統計資料

   - 最近三年來自台灣的只有 4 篇
   - 來自日本的也只有 3 篇

Thank you.