Password meters

Schwarenthorer Yannick, 1229026

June 6, 2017

Abstract

This summary is about password meters and the effectiveness of them. We will look into the details of how one of the currrent best meters (zxcvbn [2]) work and what charactaristics such a meter needs. The effect of the present of a meter of the password choosing behaviour [1]. In the paper

Contents

1 Introduction

Passwords are still the main authentication mechanism on all kinds of systems and according to the leading tech guys they will be with us for at least 10 years. The security and privacy of all our data relies on a human generated string of characters and numbers. This rich structure makes them a target of guessing attacks.

2 Comparison

In this section we will compare some password meters from commonly know websites. We will explane briefly how all of them work internally.

2.1 Evaluation Framework

Evaluation Framework –¿ - no harder than LUDS to adopt - should estimate guessing order - accurate at low magnitudes - adjustable size

3 Zxcvbn

Zxcvbn is the open sourced password meter developed from Dropbox inc. It checks how common a password is and how easily it can be guessed by an attacker. Therefore it calculates heuristically how much attempts an attacker would need to guess the password. To achieve this zxcvbn separates the work in three phases: match, estimate and search. A simple example would be a password with 2 words from top100 common password list, zxcvbn will calculate 100^2 guesses for such a password. The algorithm assumes that the attacker knows the patterns that make up the password, but not how many or in which order.

3.1 Matching

In this phase the algorithm matches patterns to a given plaintext password. It finds a set of overlapping matches. Zxcvbn will match the following patterns: tokens, reversed words, sequences, dates, repeats, keyboard patterns and bruteforce sections. For the password 'lenovo1111' the matching phase will find the patterns: lenovo (common word), eno, one (english dictionary), no (english dictionaray), on (english dictionary or reversed no), 1111(repeat), 1111(date)

Token matcher This pattern matcher lowercases the password and checks each substring against a frequency-ranked dictionary. Additionaly a transformation for common l33t-translations is made based on a l33t-Table which maps equal looking characters to there counterpart. With this approach it is possible that the matcher can detect the base password 'logitech' from the leeted version 'l0giT3CH'.

Repeat matcher The repeat matcher looks for repeated blocks of one or more characters. This matching is proceeded recursively on a winning unit, so it is also possible to identify repeated dictionary words or dates.

Keyboard matcher Here zxcvbn looks for keyboard patterns. The matcher runs throw the password and looks up the keystrokes in a graph of keyboard

layouts. This graphs represents each key on a specific keyboard layout, connecting them to there neighbours. The matcher than counts the chain length, number of turns and number of shifted characters.

Date matcher This pattern matcher looks for dates in the password. It analysis each substring of the length of 4 up to 8 characters and checks in a table for possible splits. Each split is analysed separately with some given constraints. E.g. the year is 2 or 4 digits and not in the middle, the month is between 1-12 and the day between 1 and 31 (inclusive). For example the string "201689" has 3 candidates. First 20-16-89 which is a invalid month, second 2-0-1689 where 0 is an invalid day or month and finally the best one 2016-8-9. If multiple dates are correct the one nearest to the reference year 2016 is chosen. Similar two digit years are matched to the 20th or 21th century depending on which is closer to 2016. For simplicity zxcvbn doesn't look for improper dates like the 29. Februrary on a non leap year.

3.2 Estimation

In this phase the algorithm assigns a guess attempt to each match from the previous matching phase. Zxcvbn uses following heuristic for the estimation: "if an attacker knows the pattern, how many guesses might they need to guess the instance?" [2]. For example the previous password "lenovo" is ranked 11007th in on of the used password dictionarys. Therefore it gets assigned a score of 11007 because an attacker would try it as the 11007 one. In general it is assumed that an guesser will attempt simpler more likely patterns first.

Tokens As stated above, on tokens the frequency rank in a password dictionary is used as estimation. For Revesed tokens the guesses will be doubled since the attacker has to guess both directions of each word. The result gets doubled if the password has obvious s uppercase letters (first-character, last-character or all characters). Otherwise capitalization factor is calculated width this formula:

$$\frac{1}{2}\sum_{i=1}^{min(U,L)}\binom{U+L}{i}$$

U and L are the number of upper and lowercase letters in the token. The 1/2 term converts the total guessing space to an average attempts needed.

3.3 Search

In this phase zxcvbn is searching for sequences of non-overlapping adjacent matches, so that the password is covert and total guess attempts are minimized. For example in the "lenovo1111" password the "1-1-11" date pattern gets discarded because it requires more guesses than the repeat pattern.

4 Effect of password meters

In this section we will take a look at the effect of password meters. In the paper "Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection" [1] the showed that they present of a password meter has an effect on the password strength. However this holds only for sites which the users rated as important. For lower risk websites (sites which store no personal information about the user) the password strength is not higher when a password meter is present.

References

[1] Serge Egelman et al. "Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 2379–2388. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481329. URL: http://doi.acm.org/10.1145/2470654.2481329.

[2] Daniel Lowe Wheeler. "zxcvbn: Low-Budget Password Strength Estimation". In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 157–173. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler.