POSTER: Improved Markov Strength Meters for Passwords

Harshal Tupsamudre Vijayanand Banahatti Sachin Lodha
TCS Research
Pune, India
{harshal.tupsamudre, vijayanand.banahatti, sachin.lodha}@tcs.com

ABSTRACT

Markov-based strength meters provide more accurate estimate of password strength as opposed to rule-based strength meters. However, we observed that these meters assign very high scores to slightly altered weak passwords. It is important to score modified variants of weak passwords more conservatively as the existing password cracking tools generate such guesses much more quickly. In this paper, we propose a simple greedy algorithm to detect small alterations and improve the scoring mechanism of Markov strength meters.

Keywords

Passwords; Markov Model; Strength Meter

1. INTRODUCTION

Many websites have deployed password strength meters to assist users in choosing stronger passwords. However, the impact of these meters is limited as their logic is often ad-hoc and based on very simple rules [1]. Most strength meters assign very high scores to passwords that satisfy certain length and character classes requirements. For instance, Twitter requires passwords to have a minimum length of six and to contain all character classes. Other strength meters that rely on some form of blacklisting can be easily tricked by making trivial changes to weak passwords. For instance, Yahoo does not accept "password1" but accepts "password2" as a valid password while Google classifies both these passwords as weak, it classifies "password!3" as strong. Thus, the rule-based approaches are too naive to capture the intricate user behaviour and end up allowing insecure passwords.

The data-driven approaches on the other hand provide a more valid estimate of password strength. The Markov-based strength meter proposed by Castelluccia *et al.* [2] estimates password complexity using the probabilities of ngrams that compose the given password. The Markov model exploits the fact that subsequent choices in a human-generated sequence are largely dependent on previous choices, *e.g.*, the letter 'e' is more likely to follow "th" than the letter 'z'. The n-gram Markov model predicts the next character in a sequence using previously seen $n-1$ characters. The probability of a l length character sequence $s_1 s_2 \ldots s_l$ can be therefore modelled as,

$$P(s_1 \ldots s_l) = P(s_1 \ldots s_{n-1}) \cdot \prod_{i=n}^{l} P(s_i | s_{i-n+1} \ldots s_{i-1}) \tag{1}$$

The probability of a string is often converted into *Markov score* I_{mar}, measured in bits using the formula,

$$I_{mar}(s_1 \ldots s_l) = -log(P(s_1 \ldots s_{n-1})) - \sum_{i=n}^{l} log(P(s_i | s_{i-n+1} \ldots s_{i-1})) \tag{2}$$

For demonstration purpose, we trained 4-gram Markov model on 32 million Rockyou dataset. To decide when a password should be classified as weak or strong, we computed the strength of all Rockyou passwords using Markov-based strength meter. We found that 25% of the passwords scored less than 21.14 bits (first quartile), 50% passwords scored less than 26.40 bits (median) while 75% passwords scored less than 33.12 bits (third quartile). Therefore, we classify passwords with a score below 21.14 bits as weak, between 21.15 and 33.12 bits as medium and above 33.12 bits as strong.

Password	Score I_{mar}	Strength	Edit Operation
password	11.81	weak	–
password**1**	14.29	weak	Append '1'
password**2**	16.43	weak	Append '2'
password**!3**	23.41	medium	Append "!3"
passwrd	22.30	medium	Delete 'o'
passw**1**ord	42.92	strong	Insert '1'
passw**!**ord	43.88	strong	Insert '!'
pass**ws**ord	41.71	strong	Insert 's'
pas**ws**ord	42.98	strong	Transpose "sw"
pass**v**ord	35.29	strong	Substitute 'w'

Table 1: Strength of "password" and its variants as computed by Markov-based strength meter.

Benefits. The main advantage of Markov-based strength meters is that they do not require any dictionary to operate and can be used to evaluate strings that never occurred in the training dataset. Moreover, these meters capture common password-creation strategies of users such as addition or deletion of characters at the front or end of passwords [3, 4] and score these altered passwords more reliably.
For instance, the strength of "password" as evaluated by Markov-based strength meter is 11.81 bits and the strength of its variants "password1", "password2" and "password!3" is 14.29, 16.43 and 23.41 bits respectively. All these passwords have low scores and are classified as weak or medium.
Drawbacks. Recent study [5] shows that users perceive

CCS'16 October 24-28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4139-4/16/10.
DOI: http://dx.doi.org/10.1145/2976749.2989058

adding digits and symbols in the middle of passwords as more secure than adding them at the end. Past surveys [6, 7] also suggest that placing characters in the middle of a password is not an uncommon phenomenon. We observed that although Markov-based strength meters detect insertions at the end of passwords more reliably, they assign very high scores to passwords if insertions happen in the middle. For instance, the score of "password" is just 11.81 bits but the score of its variant "passw1ord" is 42.92 bits. The increase in score by more than 30 bits just by inserting a single digit '1' is clearly an overshoot. Similar high scores are assigned if other edit operations such as substitution, transposition and deletion are performed on weak sequences such as "password" (Table 1). This is problematic because,

- It might lead users to learn inaccurate model about the password strength and cause them to practice this insecure behaviour on other critical websites too. Further, Markov-based meter assigns a score of 41.10 bits to a random looking string "jrbdlzu" and a similar score to "passw1ord" and "pawsord" both obtained by editing "password" whose score is just 11.81 bits. Strength meters should encourage the use of strings such as "jrbdlzu" and discourage trivial variants of weak passwords such as "passw1ord" and "paswsord".
- Research shows that different attack techniques generate guesses in different order [8]. Dictionary-based password crackers such as John the Ripper (JTR) and Hashcat already have rules for generating insertion, deletion, substitution and transposition of dictionary words. Consequently, these cracking tools can arrive at guesses such as "passw1ord" and "paswsord" more quickly than Markov models. Therefore, it is necessary to detect trivial changes made to weak passwords and score them more reliably.

In our work, we improve the scoring mechanism of Markov-based strength meters. More specifically, we propose a greedy algorithm to detect simple edit changes made to weak passwords and evaluate their strength more conservatively.

2. NEW SCORING MECHANISM

The Markov-based strength meter computes password score by summing the scores of all ngrams that compose the given password. The step-by-step score calculation of "password" using Markov-based strength meter is shown in Table 2. We observed that if the string is predictable then the scores of all its ngrams are fairly low as in the case of "password", but if user performs a small modification within a string then the scores of modified ngrams increases dramatically. For instance, in "passw1ord", the respective scores of ngrams "pass" and "assw" are just 0.26 and 1.04 bits, but the score of the subsequent modified ngram "ssw1" is huge 10.99 bits (Table 2). Further, the scores of modified ngrams "sw1o", "w1or" and "1ord" are also quite high. *Thus, a single edit operation causes ripple effect which increases the overall score of the password.*

We denote the input sequence by $s_1 s_2 \ldots s_l$. A simple procedure *Algo* 1 to detect a sudden surge in the score of ngram $s_{i-n+1} \ldots s_i$ is given below. The *α-surge* occurs if the score of ngram $s_{i-n+1} \ldots s_i$ is greater than α times the average score of substring $s_2 \ldots s_{i-1}$. The first character is often more informative and we do not consider it for detecting the *α-surge*. The value of parameter α can be learned, but we found that setting $\alpha = 2$ performed well and we use this value in the rest of the paper.

Algo 1 Detection of surge in the score of ngram $s_{i-n+1} \ldots s_i$

```
1: procedure isSurgeDetected($s_1 \ldots s_i$)
2: $avg = I_{mar}(s_2 \ldots s_{i-1})/(i-2)$
3:     if $I_{mar}(s_{i-n+1} \ldots s_i) > \alpha \cdot avg$ then return true
4:     else return false
5:     end if
6: end procedure
```

Now we give a greedy algorithm for detecting a single insertion, transposition, substitution and deletion within a sequence $s_1 s_2 \ldots s_l$. If the score of any ngram within the sequence increases abruptly, we check for the possible edit operation. The final score is obtained by adding the scores of original sequence and edit operation.

ngram	Score	avg
p	5.57	-
pa	2.38	2.38
pas	2.20	2.29
pass	0.26	1.64
assw	1.04	1.47
sswo	0.08	1.19
swor	0.03	1.00
word	0.25	0.89
$I_{mar}(password)$	11.81	

ngram	Score
p	5.57
pa	2.38
pas	2.20
pass	0.26
assw	1.04
ssw1	**10.99**
sw1o	**10.11**
w1or	**5.38**
1ord	**4.99**
$I_{mar}(passw1ord)$	42.92

Table 2: Strength computation of "password" and "passw1ord" using Markov-based strength meter.

Insertion. To verify if *α-surge* in the score of ngram $s_{i-n+1} \ldots s_i$ occurred due to insertion of character s_i, we compute the score of skip-gram $s_{i-n+1} \ldots s_{i-1} s_{i+1}$. If the score of this skip-gram is low, we can infer that the character s_i is inserted in the original sequence $s_1 \ldots s_{i-1} s_{i+1} \ldots s_l$. The revised score of the sequence $s_1 s_2 \ldots s_l$ is thus the sum of score of original sequence and score due to insertion. One approach to score insertion is to consider only the character class γ of insertion. There are four such character classes, lowercase, uppercase, digit and symbol. As there are l positions for insertions, the score due to insertion is $log_2(l \cdot |\gamma|)$.

$$I_{ins}(s_1 \ldots s_l) = I_{mar}(s_1 \ldots s_{i-1} s_{i+1} \ldots s_l) + log_2(l \cdot |\gamma|) \quad (3)$$

For instance, in "passw1ord", the average score of string "passw" is just 1.47 bits but the score of the subsequent ngram "ssw1" is 7.48 times high (10.99 bits). This abrupt change in ngram score triggers our greedy algorithm which verifies if the digit '1' in "ssw1" is insertion. As the score of skip-gram "sswo" is just 0.08 bits, the algorithm recognizes '1' as insertion. Further, the character class of insertion is digit ($|\gamma| = 10$), hence the revised score of "passw1ord" is $I_{mar}(password) + log_2(9 \cdot 10) = 11.81 + 6.49 = 18.30$ bits. As the score is low, "passw1ord" is classified as weak.

Transposition. To verify if *α-surge* in the score of ngram $s_{i-n+1} \ldots s_i$ is due to transposition of characters s_i and s_{i+1}, we compute the scores of skip-gram $s_{i-n+1} \ldots s_{i-1} s_{i+1}$ and ngram $s_{i-n+2} \ldots s_{i-1} s_{i+1} s_i$. If these scores are low, we can infer that the string $s_1 \ldots s_i s_{i+1} \ldots s_l$ is obtained by transposing characters s_i and s_{i+1} in a original sequence $s_1 \ldots s_{i+1} s_i \ldots s_l$. Note that insertion is detected just by evaluating the skip gram $s_{i-n+1} \ldots s_{i-1} s_{i+1}$. As there are $l-1$ choices for transposing two consecutive characters in a l length sequence, we increase the score by $log_2(l-1)$ bits.

$$I_{trans}(s_1 \ldots s_l) = I_{mar}(s_1 \ldots s_{i-1} s_{i+1} s_i \ldots s_l) + log_2(l-1) \quad (4)$$

For instance, in "paswsord", the score changes abruptly at the ngram "pasw" (6.97 bits). As the respective scores of skip-gram "pass" and ngram "assw" are just 0.26 and 1.04 bits, we classify this operation as transposition. The corrected score of "paswsord" is $11.81 + log_2(8-1) = 14.62$ bits. Hence, "paswsord" is classified as weak.

ngram	Score
p	5.57
pa	2.38
pas	2.20
pasw	**6.94**
asws	**7.89**
swso	**5.07**
wsor	**7.88**
sord	**5.05**
$I_{mar}(paswsord)$	42.98

ngram	Score
p	5.57
pa	2.38
pas	2.20
pass	0.26
assv	**11.52**
ssvo	**4.16**
svor	**3.42**
vord	**5.78**
$I_{mar}(passvord)$	35.29

Table 3: Strength computation of "paswsord" and "passvord" using Markov-based strength meter.

Substitution. To verify if α-*surge* in the score at the i^{th} position is due to substitution operation, we fetch the top $k = 3$ frequent ngrams $s_{i-n+1} \ldots s_{i-1}s_a$, $s_{i-n+1} \ldots s_{i-1}s_b$ and $s_{i-n+1} \ldots s_{i-1}s_c$ from the training dataset. Next, we sum the scores of $s_{i-n+1} \ldots s_{i-1}t$ and $s_{i-n+2} \ldots s_{i-1}ts_{i+1}$, where $t \in \{s_a, s_b, s_c\}$ and return the minimum. The intuition is that if s_i is a replacement of t then the scores of consecutive ngrams $s_{i-n+1} \ldots s_{i-1}t$ and $s_{i-n+2} \ldots s_{i-1}ts_{i+1}$ are low. One approach to score substitution is to consider only the character class γ of substitution.

$$I_{sub} = log_2(|\gamma|) + \underset{t \in \{s_a, s_b, s_c\}}{\text{argmin}} I_{mar}(s_1 \ldots s_{i-1}ts_{i+1} \ldots s_l) \tag{5}$$

In "passvord", the score of ngram "assv" is 11.52 bits (Table 3). To detect if letter "v" is a replacement, we fetch the top 3 ngrams "assw", "assi" and "assy" from the Rockyou dataset. The scores of these ngrams are 1.04, 3.05 and 4.12 bits respectively. The scores of the subsequent ngrams "sswo", "ssio" and "ssyo" are 0.08, 3.74 and 3.48 bits respectively. As the sum of scores of ngrams "assw" and "sswo" is just 1.12 bits, we identify letter 'v' as a replacement of letter 'w'. Therefore, the revised score of "passvord" is $11.81 + log_2(26) = 16.51$ bits. Hence, "passvord" is weak.

ngram	Score
p	5.57
pa	2.38
pas	2.20
pass	0.26
assw	1.04
sswr	**9.15**
swrd	**1.70**
$I_{mar}(passwrd)$	22.30

ngram	Score
$I_{mar}(password)$	11.81
ord!	**6.89**
rd!3	**4.71**
$I_{mar}(password!3)$	23.41

Table 4: Strength computation of "passwrd" and "password!3" using Markov strength meter.

Deletion. The detection of deletion operation is similar to substitution. If the score at i^{th} position is high, we fetch the top 3 ngrams $s_{i-n+1} \ldots s_{i-1}s_a$, $s_{i-n+1} \ldots s_{i-1}s_b$ and $s_{i-n+1} \ldots s_{i-1}s_c$. In this case, we sum the ngram scores of $s_{i-n+1} \ldots s_{i-1}t$ and $s_{i-n+2} \ldots s_{i-1}ts_i$, where $t \in \{s_a, s_b, s_c\}$ and return the minimum. The intuition is that if the character t was originally deleted then the scores of $s_{i-n+1} \ldots s_{i-1}t$ and $s_{i-n+2} \ldots s_{i-1}ts_i$ are low. We assume that a character can be deleted from any of the $l+1$ positions and add $log_2(l+1)$ bits to the score of original sequence.

$$I_{del} = log_2(l+1) + \underset{t \in \{s_a, s_b, s_c\}}{\text{argmin}} I_{mar}(s_1 \ldots s_{i-1}ts_{i+1} \ldots s_l) \tag{6}$$

In "passwrd", the score of ngram "sswr" is very high 9.15 bits (Table 4). We fetch the top 3 ngrams "sswo", "ssw0" and "sswi". As the sum of scores of ngrams "sswo" and "swor" is just 0.11 bits, we infer that the letter 'o' was deleted. Thus, "passwrd" score is just $11.81 + log_2(8) = 14.81$ bits.

Now we provide a pseudocode of our greedy algorithm. The objective is to detect changes made to weak passwords and to score them more conservatively. For every position i in password $s_1 \ldots s_l$, we invoke *Algo* 2 which explores possible edits and returns score I_i. If the algorithm overestimates the strength of a password string, we assign the original score I_{mar} as computed by the Markov-based strength meter. Therefore, the final score is,

$$I_{new}(s_1 \ldots s_l) = min(I_{mar}, I_1, I_2, \ldots, I_l) \tag{7}$$

Algo 2 Detection of edit operation in ngram $s_{i-n+1} \ldots s_i$

```
1:  procedure EditDetection($s_1 \ldots s_l$, $i$)
2:      if !isSURGEDETECTED($s_1 \ldots s_i$) then return $I_i = \infty$
3:      end if
4:      $avg = I_{mar}(s_2 \ldots s_{i-1})/(i-2)$
5:      $skip = I_{mar}(s_{i-n+1} \ldots s_{i-1}s_{i+1})$
6:      if $skip \leq \alpha \cdot avg$ then
7:          $I_{ins} = I_{mar}(s_1 \ldots s_{i-1}s_{i+1} \ldots s_l) + log_2(l \cdot |\gamma|)$
8:          if $I_{mar}(s_{i-n+2} \ldots s_{i-1}s_{i+1}s_i) \leq \alpha \cdot avg$ then
9:              $I_{trans} = I_{mar}(s_1 \ldots s_{i-1}s_{i+1}s_i \ldots s_l) + log_2(l-1)$
10:         end if
11:     end if
12:     ▷ Steps 13 to 21, try for all characters t and return minimum
13:     $top = I_{mar}(s_{i-n+1} \ldots s_{i-1}t)$
14:     $substitute = top + I_{mar}(s_{i-n+2} \ldots s_{i-1}ts_{i+1})$
15:     if $substitute \leq 2 \cdot \alpha \cdot avg$ then
16:         $I_{sub} = I_{mar}(s_1 \ldots s_{i-1}ts_{i+1} \ldots s_l) + log_2(|\gamma|)$
17:     end if
18:     $delete = top + I_{mar}(s_{i-n+2} \ldots s_{i-1}ts_i)$
19:     if $delete \leq 2 \cdot \alpha \cdot avg$ then
20:         $I_{del} = I_{mar}(s_1 \ldots s_{i-1}ts_{i+1} \ldots s_l) + log_2(l+1)$
21:     end if
22:     return $I_i = min(I_{ins}, I_{trans}, I_{sub}, I_{del})$
23: end procedure
```

The algorithm can be adapted to detect more than one changes. However, our intent is not to score stringently if user performs multiple changes to their password as it increases the effort of an attacker. Also, we observed that the effect of edits happening at the beginning is less pronounced, *e.g.*, the strength of "p1assword" is just 26.72 bits.

3. CONCLUSION

Users often circumvent the enforced policies by making trivial changes to weak passwords. The attacker can try many variations of weak passwords before generating the next set of guesses. Therefore, it is important to detect alterations made to weak passwords and score them more conservatively. We demonstrated that although Markov-based strength meters are very sensitive to small tweaks made to weak passwords, the sudden surge in ngram scores can be used to detect these modifications. The score of a password is then the sum of score of the original sequence and score of the tweak. This improved scoring technique allows us to estimate the strength of passwords more reliably.

4. REFERENCES

[1] Xavier de Carné de Carnavalet et al. From very weak to very strong: Analyzing password-strength meters. In *NDSS*, 2014.

[2] Claude Castelluccia et al. Adaptive password-strength meters from markov models. In *NDSS*, 2012.

[3] Richard Shay et al. Encountering stronger password requirements: User attitudes and behaviors. In *SOUPS*, 2010.

[4] Blase Ur et al. "I added '!' at the end to make it secure": Observing password creation in the lab. In *SOUPS*, 2015.

[5] Blase Ur et al. Do users' perceptions of password security match reality? In *CHI*, 2016.

[6] Saranga Komanduri et al. Of passwords and people: Measuring the effect of password-composition policies. In *CHI*, 2011.

[7] Anupam Das et al. The tangled web of password reuse. In *NDSS 2014*.

[8] Blase Ur et al. Measuring real-world accuracies and biases in modeling password guessability. In *USENIX Security Symposium*, 2015.