PASSTRANS: AN IMPROVED PASSWORD REUSE MODEL BASED ON TRANSFORMER

Xiaoxi He[¶] *Haibo Cheng*[¶*] *Jiahong Xie*[¶] *Ping Wang*[¶*] *Kaitai Liang*[§]

[¶]Peking University
{hxx2019, hbcheng, xjhshare, pwang}@pku.edu.cn
[§]Delft University of Technology
kaitai.liang@tudelft.nl

ABSTRACT

Passwords have been widely used in online authentication, and they form the front line that protects our data security and privacy. But the security of password may be easily harmed by insecure password generator. Massive reports state that users are always keen to generate new passwords by reusing or fine-tuning old secrets. Once an old password is leaked, the users may suffer from credential tweaking attacks. We propose a password reuse model PassTrans and simulate credential tweaking attacks. We evaluate the performance in leaked password datasets, and the results show that 67.51% of accounts is breakable under 1,000 guesses, indicating our model is accurate in capturing password reuse behavior.

***Index Terms*—** credential tweaking attack, password model, password reuse, similarity

1. INTRODUCTION

Online authentication is of extreme importance in safeguarding valid Internet users' resource and service access rights. There have been various authentication methods so far [1, 2]. But password-based approach is still the mainstream due to its advantages on fast deployment and handy usability. Password may be regarded as the front-line protection for Internet users' digital assets and meanwhile, its security may easily get on our nerves [3]. Reports said that users always keen to reuse or slightly modify the old password to generate new credentials due to their limit human memory [4]. This fragile password generation may bring potential leakage risk to the new passwords once the old credentials are compromised, which is known as credential tweaking attack. The attack is illustrated in Fig. 1. Specifically, an attacker collects the leaked accounts from websites A, then launches possible guessing by tweaking the obtained passwords, and finally attacks website B by login attempts.

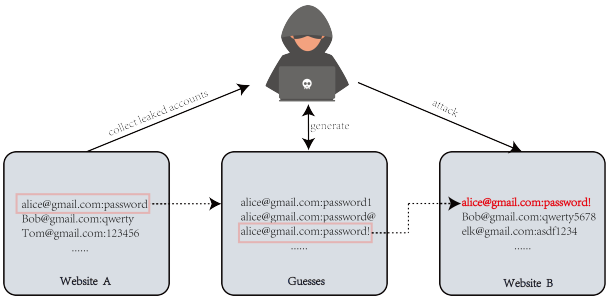


Fig. 1. Credential tweaking attack

In 2014, Das et al. [5] first studied the threat of password reuse and proposed a cross-site password guessing algorithm. They assumed that a user's new password can be generated by modifying its old password via some transformation rules, such as *reverse* (e.g., "123456" $\rightarrow$ "654321"). But the priority of the rules is fixed in the algorithm and they did not consider mixing multiple rules. In 2016, Wang et al. [6] proposed Targuess II, which is also a cross-site password guessing model based on transformation rules. They trained the probability of each rule to obtain the probability of a guessing password. In their design, one can attack a user's account according to the decreasing probability. Later on, Pal et al. [7] introduced a password similarity model based on deep learning: Pass2path. The output of their model is the transformation path of leaked passwords to targeted passwords. However, there are multiple transformation paths for a pair of passwords, in which one of them may be invalid. As a result, the number of unique passwords does not match to transformation path, which limits their attack performance.

The aforementioned models may not yet practical enough to be used. This is so because their reuse rules are defined artificially and limited; and standing at the viewpoint of password users, multiple "unrelated" passwords (e.g., "chixue100" $\rightarrow$ "123456") could be used in practice, which breaks the "dependency" on the previous assumption. To tackle the issues, we propose PassTrans, a credential tweaking attack model based on transformer [8].

*Corresponding author.

This research is supported by National Key R&D Program of China (2020YFB1805400), China Postdoctoral Science Foundation National (2021M700215), Natural Science Foundation of China (62072010) and European Union's Horizon 2020 research and innovation programme under grant agreement No. 952697 (ASSURED) and No. 101021727 (IRIS).

1.1. Our Contributions

We use a neural network to automatically extract the transformation rules. In order to obtain visible feature extraction capability, we build our model based on transformer [8], which is able to deliver great performance in machine translation tasks. In our model - PassTrans, we feed the leaked passwords directly into the model, and obtain the guessing without transformation rules. PassTrans is trained on 4iQ [3] which contains billions of passwords. To evaluate the fitting ability of PassTrans on password reuse behavior, we simulate attacks on practical datasets. And the experiments show that PassTrans can crack 67.51% of passwords under 1,000 guesses, which outperforms Pass2path,

2. PASSWORD REUSE MODEL

Passwords have been massively reused on different websites [4]. When attackers compromise a password that a user has used, they may use this password as a step-stone, by modifying it, to attempt login to the user's other accounts. We call this credential tweaking attack. Das et al. [5] proposed a heuristic credential tweaking attack algorithm based on transformation rules. The algorithm transforms the leaked password in sequence - sequential operations (such as "abc" followed by 'd'), deletion, insertion, uppercase, and a series of artificially defined rules - and then outputs the guessed password. But the shortcomings are clear there. The priority of the transformation rules is fixed, which leads to generating many inreasonable guessing. For example, the transformation rule of "password" $\rightarrow$ "pa$$word" is *leet*; but *reverse* is ranked before *leet*, the method then will generate "drowssap" first. Furthermore, their design does not consider the mixed rules of multiple transformation. Later on, Wang et al.[6] proposed Targuess II which assumes that users modify old passwords by fixed rules (such as adding or deleting a character, adding or deleting a structure, special transformation, etc.). They analyzed the transformation path of the password pair by editing distance, via a series of probabilities of transformation rules, and generated the guesses with the conditional probability decreasing. They took mixed rules into consideration and selected the rules according to probability. But the transformation rules they used are still artificially defined without considering irrelevant passwords.

In 2019, Pal et al. [7] proposed Pass2path. They first adapted deep learning to construct model instead of statistical learning. Their goal is to predict the generation path of the targeted password (note the generation path is the sequence of artificially defined transformation rules). For example, the path from "cats" to "kates" is: *(sub,k, 0), (ins,e,3)*, which means we can substitute 'c' for 'k' first, and then insert 'e' at the third character of "cats", and finally get "kates". There may be multiple paths for password-to-password transformation, but only one transformation path is used for training.

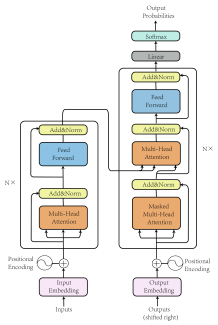


Fig. 2. Transformer [8]

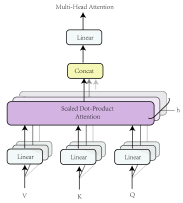


Fig. 3. Multi head attention [8]

This brings difficulty to determine which one should be chosen to capture better characteristics under different sceneries. For instance, there are two paths for transformation between "123456" and "12345678". The first one is to add '8' before inserting '7' between '6' and '8', while the second option is to add '7' and then '8'. Besides, the model may generate invalid generation path, such as *(sub,k, 7), (ins,e,8)* for "cats" to "kates". And again, their transformation rules only cover relevant passwords.

3. OUR PROPOSED METHOD

In order to characterize a user's password reuse behavior, such as direct reuse (e.g., "password" $\rightarrow$ "password"), fine-tuning (e.g., "password" $\rightarrow$ "pa$$word") and irrelevant passwords (e.g., "chixue100" $\rightarrow$ "123456"), we propose PassTrans based on the transformer structure.

3.1. Transformer

In 2017, Vaswani et al. [8] developed the transformer model, which provides great performance in machine translation and has been widely used in many fields [9, 10]. They proposed multi-head attention to improve the feature extraction ability of the network. The transformer is designed as the encoder-decoder architecture. The encoder converts the sequence into hidden representation, while the decoder converts the hidden representation into output sequence. Unlike the seq2seq model presented by Sutskever et al. [11], the transformer does not adopt RNN [12] and LSTM [13]; instead, it leverages a new structure including position embedding, multi-head attention, and feed forward network modules, as shown in Fig.2.

3.1.1. Scaled Dot-Product Attention

Bahdanau et al. [14] proposed an attention model on the machine translation tasks. The attention model (with information extraction function) simulates human language mechanism, and only pays attention to the key information rather than the

useless. Scaled Dot-Product Attention mechanism is the basic module of the transformer. Assume that the query matrix of a given task is **Q**, the input information is represented by **K-V**, which is used to calculate the correlation score with **Q**. Then we have

$$\text{Attention}\,(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{4iQ_k}}\right)V \qquad (1)$$

where $4iQ_k$ is the dimension of the input vector and **K** represents the key.

3.1.2. Multi-Head Attention

Multi-head attention mechanism is the core module of the transformer. Its core idea is to calculate the attention multiple times and concatenate it. In Fig.3, *h* represents the number of headers. The larger *h* we set, the stronger the feature extraction ability we have. In our setting, we define *h*=4.

3.2. The Design of PassTrans

3.2.1. Building PassTrans

Based on the transformer [8], we propose a credential tweaking attack model, which is called PassTrans. Assume the leaked password is pw_{leak}, the password to guess is pw_{guess}. Our purpose is to model the conditional probability $P(pw_{guess}|pw_{leak})$ given as follows.

$$\begin{aligned}P(pw_{guess}|pw_{leak}) &= P\left(c_1,\ldots,c_l \mid \hat{c}_1,\ldots,\hat{c}_{l'}\right)\\ &=\prod_{i=1}^{l} P\left(c_i \mid \hat{c}_1,\ldots,\hat{c}_{l'},c_1,\ldots,c_{i-1}\right)\end{aligned} \qquad (2)$$

Different from natural language processing, passwords focus on characters instead of words. Therefore, our dictionary collection is printable characters. As shown in Fig.4, PassTrans consists of encoder and decoder. Firstly, we put the leaked password into the encoder, and then get the output of the encoder, denoted as en_{out}. Further, we can obtain the first character c_1 by entering the start character τ and en_{out}; and at the same time, we obtain the conditional probability $P\left(c_1 \mid \hat{c}_1,\ldots,\hat{c}_{l'}\right)$, c_2 and $P\left(c_2 \mid \hat{c}_1,\ldots,\hat{c}_{l'},c_1\right)$ are generated by entering en_{out} and τ, c_1 into the decoder. We then keep iterating until the end character ϵ is generated. Finally, we have the pw_{guess} and $P(pw_{guess}|pw_{leak})$ by multiplying the above conditional probabilities.

In our model, both encoder and decoder adopt three-layer transformer structure. The number of head is set to 4 in the multi-head attention module, and the dimension is 64. The feedforward network consists of two layers of one-dimensional convolutional networks. Due to the limit of users' memory, passwords are usually short. As in [6, 7], we restrict the max length of a password to 30.

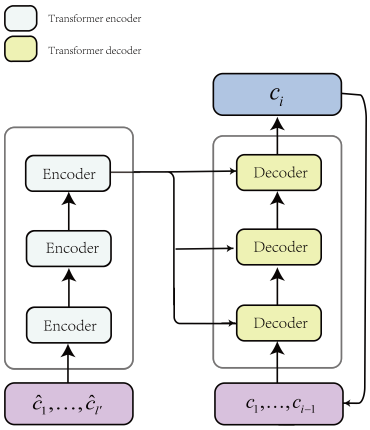


Fig. 4. Generate c_i by PassTrans

Table 1. Password dataset information

Dataset	Users	Total Passwords	Unique passwords
4iQ	1.46×10^9	3.60×10^9	1.72×10^9
Pastebin	276	2272	739

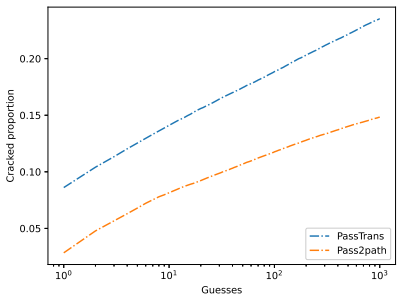
3.2.2. Generating Guesses

In order to generate multiple guesses, PassTrans adopts beam search technology [15], which is applied in natural language process. We assume that the number of guesses is q, and our purpose is to generate the most likely q guesses, i.e. the top q of the probability $P(pw_{guess}|pw_{leak})$. It may take a lot of time to traverse all possible guesses and then select those with the top q. Fortunately, beam search is an approximate solution that maintains a balance of time and performance. In the process of generating guesses, we 1) select the top q paths at each iteration; 2) iterate until the end character appears, then we say that a guess is generated; and 3) output q guesses. This algorithm leverages the greedy algorithm and prunes the spanning tree, which significantly improves the generation efficiency.

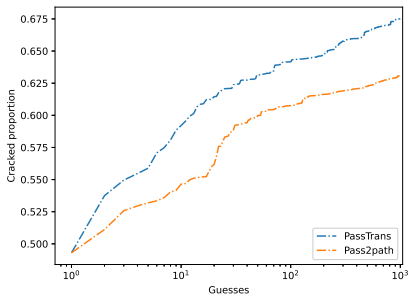
There are some shortcomings on beam search that can be improved. First, beam search may miss the optimal sequence because it is a greedy algorithm; in addition, since the longer the sequence, the lower the score, it tends to generate short sequences, it would be improved adding length normalization and coverage penalty [16].

4. EXPERIMENTS AND RESULTS

The experimental results provided by Pal et al. [7] show that the performance of Pass2path is much better than the approach designed by Das et al. [5] and Targess II [6]. Thus we here only compare our design with the best - Pass2path. The model parameter settings of Pass2path are consistent with the public model [17].



(a) 4iQ_{ts}



(b) Pastebin

Fig. 5. Percentage of guessed password under 1000 guesses on different datasets

4.1. Password Datasets

We use two public password datasets to present practical attack results, and their details are given in Table 1. One is the password set collected from 4iQ [3], which contains 1.4 billion email password pairs, and the duplicate entries are removed. The other dataset is Pastebin [18], which is widely used in the research of password vault [18, 19]. And it includes 276 user password sets.

4.2. Experiment Setup

In 4iQ, to obtain the user's password set, we use the same method as in Pal et al. [7]. Specifically, passwords with the same email are aggregated as a password set. In addition, we only retain users with passwords ranging from 2 to 100, which is reasonable in practical applications. Through this processing, we have 146 million users. In our setup, 80% of the dataset is used as the training set, and the remaining 20% is for testing, which are recorded as $4iQ_{tr}$ and $4iQ_{ts}$, respectively.

Our goal is to successfully guess the targeted password through the leaked password under a certain number of guesses. Therefore, we take the number of guesses as our experimental overhead and record it as q. In our experiment, 1000 guesses are generated and sorted by decreasing probability. In order to improve the experimental efficiency, we randomly select one hundred thousand users from the $4iQ_{ts}$, and then select a pair of passwords as the leaked password and the targeted password respectively for each user. As for Pastebin, we first select the leaked password as a starting point, and then use PassTrans to fine tune $q-1$ passwords for guessing.

4.3. Analysis of Experimental Results

The experimental results are shown in Fig.5, Table 2 and Table 3. From the results obtained from 4iQ_{ts}, we find that the performance of PassTrans is increased by 57.2% – 62.7%, as compared to Pass2path. In Pass2path, there are multiple paths for generating a target password, so that some guessed passwords are repeated. This makes the actual guessing number less than q. Besides, Pass2path doesn't consider irrelevant passwords. In the attacks against Pastebin, we can see that the guessing success rate of the two models is greater than 50%. This is reasonable in daily life, since some users may reuse their old passwords without any modification. PassTrans outperforms Pass2path in all experimental metrics. Under 1000 guesses, PassTrans reaches approx. 67.51% cracking success rate.

In our experiment, the user's password dataset is aggregated by email names, not the user's real password set. It may be a better choice that simulating credential tweaking attack experiments in real life.

Table 2. Percentage of guessed password on 4iQ_{ts}

Attack Method	$q = 10$	$q = 10^2$	$q = 10^3$
Pass2path	8.07	11.71	14.83
PassTrans	**13.13**	**18.41**	**23.52**

Table 3. Percentage of guessed password on Pastebin

Attack Method	$q = 10$	$q = 10^2$	$q = 10^3$
Pass2path	54.15	60.72	63.07
PassTrans	**58.81**	**63.93**	**67.51**

5. CONCLUSION

We introduce PassTrans - a password reuse model based on transformer for credential tweaking attack. PassTrans uses the transformer to build neural network, avoiding to generate passwords via the generation path approach. We perform experiments on two real password datasets, 4iQ_{ts} and Pastebin. The results show that PassTrans achieves better password guessing performance than Pass2path. There are some open problems, such as the explanation of password reuse behavior, password strength evaluation and et al..

6. REFERENCES

[1] Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE PAMI*, vol. 19, no. 7, pp. 711–720, 1997.

[2] Anil Jain, Lin Hong, and Ruud Bolle, "On-line fingerprint verification," *IEEE PAMI*, vol. 19, no. 4, pp. 302–314, 1997.

[3] Julio Casal, "1.4 billion clear text credentials discovered in a single database," `https://medium.com/4iqdelvedeep/1-4-billion-clear-textcredentials-discovered-in-a-single-database-3131d0a1ae14`, 2017.

[4] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot, "An administrator's guide to internet password research," in *Proc. LISA14*, 2014, pp. 44–61.

[5] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang, "The tangled web of password reuse.," in *Proc. NDSS*, 2014, vol. 14, pp. 23–26.

[6] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS*, 2016, pp. 1242–1254.

[7] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P*. IEEE, 2019, pp. 417–434.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[9] Hari Krishna Vydana, Martin Karafiát, Katerina Zmolikova, Lukáš Burget, and Honza Černocký, "Jointly trained transformers models for spoken language translation," in *Proc. ICASSP*, 2021, pp. 7513–7517.

[10] Xuankai Chang, Wangyou Zhang, Yanmin Qian, Jonathan Le Roux, and Shinji Watanabe, "End-to-end multi-speaker speech recognition with transformer," in *Proc. ICASSP*, 2020, pp. 6134–6138.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[15] Christopher Makoto Wilt, Jordan Tyler Thayer, and Wheeler Ruml, "A comparison of greedy search algorithms," in *third annual symposium on combinatorial search*, 2010.

[16] Yonghui Wu, Mike Schuster, and et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[17] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart, "Password similarity models using neural networks," `https://github.com/Bijeeta/credtweak`, 2019.

[18] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart, "Cracking-resistant password vaults using natural language encoders," in *Proc. IEEE S&P*. IEEE, 2015, pp. 481–498.

[19] Maximilian Golla, Benedict Beuscher, and Markus Dürmuth, "On the security of cracking-resistant password vaults," in *Proc. ACM SIGSAC*, 2016, pp. 1230–1241.