GENPass: A General Deep Learning Model for Password Guessing with PCFG Rules and Adversarial Generation

Yunyu Liu*, Zhiyang Xia*, Ping Yi*, Yao Yao†, Tiantian Xie†, Wei Wang†, Ting Zhu†
*Shanghai Key Laboratory of Integrated Administration Technologies for Information Security,
School of Cyber Security, Shanghai Jiao Tong University, Shanghai, 200240, China
†Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD, 21250, USA

Abstract—**Password has become today's dominant method of authentication in social network. While the brute-force attack methods, such as HashCat and John the Ripper, are unpractical, the research then switches to the password guess. The state-of-the-art approaches, such as Markov Model and probabilistic context-free grammars(PCFG), are all based on statistical probability. These approaches have a low matching rate. The methods on neural network have been proved more accurate and practical for password guessing than traditional methods. However, a raw neural network model is not qualified for cross-sites attack since each data set has its own features.**

This paper proposes a general deep learning model for password guessing, called GENPass. GENPass can learn features from several data sets and ensure the output wordlist high accuracy in different data sets by using adversarial generation. The password generator of GENPass is PCFG+LSTM(PL), where LSTM is a kind of Recurrent Neural Network. We combine neural network with PCFG because we found people were used to set their passwords with meaningful strings. Compared with LSTM, PL increased the matching rate by 16%-30% in the cross-sites tests when learning from a single dataset. GENPass uses several PL models to learn datasets and generate passwords. The result shows that the matching rate of GENPass is 20% higher than that of simply mixing those datasets in the cross-sites test.

I. Introduction

Deep learning [1] astonished the world after AlphaGo [2] won Lee Sedol. The result proves that computers can emulate and surpass human-beings after establishing a suitable model. Therefore, we try to explore a method to apply deep learning to password guessing in this paper.

Many researches, as well as open source tools, aim to generate a wordlist with a high matching rate to match real passwords. HashCat [3] and John the Ripper(JtR) [4] are two remarkable password cracking tools, but they can only generate a finite wordlist, which is far from enough for users to crack most of the passwords. Markov models [5] and probabilistic context-free grammars [6] are widely-used techniques for password guessing. Both of them are based on the probability model. Therefore, they are often computationally intensive and time-consuming [7]. Melicher et al. [8] first introduced neural network to guess passwords and Hitaj et al. [9] recently proposed passGAN, using GAN to increase the accuracy. However, both Melicher et al. and Hitaj et al. only focused on one-site test. For example, RockYou is both used in training and testing dataset. As we all know, cross-sites attack is the common method to crack a database. Although their performances are good, those generated password lists are not convinced to be general.

To solve these problems, we proposed a new general model called GENPass. GENPass can enhance both the accuracy and the generality. The accuracy is improved by using PCFG+LSTM(PL) model, the generator of GENPass. This model compromises the merits of PCFG and LSTM models. We use PCFG rules to replace sequences with tags. The encoded sequence is named as tagged-sequence in this paper. These tagged-sequence were fed to a LSTM network for training and generation. LSTM network has been proved good at password guessing [8]. By using PCFG rules, the number of guesses can be reduced 5-6 orders of magnitude when achieving 50% matching rate. The generality means that the output wordlist can achieve a relatively high matching rate in different datasets. We proposed GENPass to improve the generality. GENPass implements the adversarial idea in the process of generation. It enables the model to generate a wordlist based on several sources. Therefore, the new generated wordlist is more eclectic. The model consists of several generators and a classifier. Generators create passwords from datasets, namely leaked passwords, and the task of classifier is to make sure the output does not belong to a certain dataset so that the output is believed to be general to all the training datasets.

The contributions of this paper are as follows.

- PL can significantly improve the matching rate of LSTM in both one-site and cross-sites tests.
- Based on GENPass, we generate a general wordlist by learning several leaked password datasets. A general wordlist means the matching rate of the wordlist is relatively high in cross-sites tests.

The remainder of this paper is structured as follows. Section II briefly introduces traditional approaches and previous researches in password guessing field. Section III describes the model design of GENPass. In Section IV, we evaluate GENPass, on the comparison with state-of-art methods and other neural network models. We conclude the paper and discuss further work in Section V.

II. Related Works

In the previous section, we do some research on network attack and intrusion detection [10]–[22]. In this section, we first describe traditional approaches to crack password and analyze their disadvantages. Then we introduce the recent development of deep learning and its usage in password guessing area.

A. Traditional Password Guessing Methods

In dictionary attack, we know a target hashed passwords. Then we calculate the hash code of each passwords in the dictionary and compare it with the target one. HashCat and JtR is popular because they can greatly accelerate the hash code computing. However, such password crackers must be based on a given password wordlist. The size of the wordlist means the upper limit for one cracking attempt.

Markov model was used for password guessing in 2005 by Narayanan et al. [23] for the first time and improvement has been proposed recently [5]. The core concept of Markov model is time-space tradeoff, using huge extra space to calculate the probability of the next character based on previous or context ones.

Probabilistic context-free grammar (PCFG) was introduced in password guessing in 2009 by Weir et al. [6]. PCFG derives from word-mangling rules based on a training set of leaked passwords. The grammars then generate passwords with trained probabilistic model. The regularity [24] of a dataset is preserved by encoding these passwords with PCFG rules. PCFG rules will be specified in the Section III. This approach was extended to targeted password guessing [25], [26], which means the grammar is not only at character level, but also personally identifiable information(PII).

Li et al. [24] has also proved that there is a great difference between English passwords and Chinese ones. According to their statistical data, Chinese prefer digits and Pinyin while English users prefer letters when setting passwords. So our general model only focuses on one language environment.

B. Neural Network in Password Guessing

Neural network was first used in password guessing by Melicher et al. [8] in 2016. LSTM [27], a recurrent neural network, generates one character at a time, similar to Markov model, but the neural network method does not cost any extra space. The evaluation of [8] shows that wordlist generated by neural network outperforms those of Markov model, PCFG, HashCat and JtR, particularly at the guessing number above 10^{10}. However, [8] restricts the structure of password(e.g., 1class8, 3class12), so the result cannot be considered general. PassGAN proposed by Hitaj et al. [9] used Generative Adversarial Network(GAN) to generate passwords. GAN provides an adversarial process to make artificial passwords more real. With the help of GAN, PassGAN was able to create passwords that traditional password generation tools could not create.

III. The Model of GENPass

In this section, we describe the design of the PCFG+LSTM (PL) and the GENPass model in details.

A. PCFG+LSTM (PL)

A regular password guessing method called probabilistic context-free grammars (PCFG) [6] divides the password by unit. For instance, 'password123' can be divided into two units 'L8' 'D3'. This brings a high accuracy because the passwords are always meaningful and set with template structures (e.g., iloveyou520, password123, abc123456). Meanwhile, the neural networks can detect the relationship between characters that PCFG cannot. Thus, we combine two methods to get a more effective one.

1) Preprocessing: A password is first encoded into a sequence of units. Each unit has a char and a number. A char stands for a sequence of letters(L), digits(D), special chars(S) or end character('\n') and the number stands for the length of the sequence (e.g., $Password123 will be donated as S1 L8 N3 '\n'). Detailed transform rules are shown in table I.

TABLE I
DIFFERENT STRING TYPES

Data Type	Symbols
Letters	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
Digits	0123456789
End character	\n
Special Chars	otherwise

2) Generating: The long short-term memory (LSTM) model is a deformation of the basic RNN model. We use LSTM to generate passwords. By feeding the LSTM model the preprocessed wordlist and training, the model can predict the next unit.

3) Guessing Password: When a unit is determined, it will be transformed back into an alphabetic sequence. This sequence can be in different forms. Each form appears (e.g., L4 can be expressed as 'ever', 'love', 'life' ...) in different frequencies in the wordlist (e.g., there are 47 'ever', 146 'love', 105 'life'...). The definite alphabetic sequence will be determined by a weight choosing process.

4) Weight Choosing: Experiments have shown that if the highest weight output is chosen each time, there will be a large number of duplicates in the output wordlist. Random selection can ensure that higher weight candidates are chosen at a higher probability, while the lower ones can still be chosen after a large number of guesses. We design a random selection algorithm. Assume $t[i](i = 1...n)$ is the frequency of the $i - th$ candidate. The total frequency is $s = \Sigma_{i=1}^{n} t[i]$. $[0, s)$ which can be divided into n regions. The $i - th$ region stands for $i - th$ candidate. The first region is $[0, t[1])$. The $i - th(2 \leq i \leq n)$ region is $[\Sigma_{j=1}^{i-1} t[i], \Sigma_{j=1}^{i} t[i])$. Generate a random number p between 0 and s. The chosen one's region should include p. For example, the frequencies of candidates are ['abcd':4,'love':3,'life':5]. The region of 'abcd' is [0,4), the region of 'love' is [4,7) and the region of 'life' is [7,12). s is 12. If the random number p is 5, the chosen string is 'love'. If the random number p is 10, the chosen string is 'life'. The pseudo code is shown in Algorithm 1.

Algorithm 1: Weight Choosing

```
1 $t[0] = 0$;
2 $t[i] = \Sigma_{j=1}^{i} weights[j] (i = 1...n)$;
3 $s = \Sigma_{i=1}^{n} weights[i]$;
4 $index = 0$;
5 p is a random value from 0 to $s$;
6 repeat
7     index = index + 1
8 until $t[index] > p$ or $index = n$;
9 return $index$;
```

B. GENPass

GENPass generates a general wordlist from different datasets. Because different datasets have different principles and different lengths, it is hard to learn the general principles by simply mixing them. To solve that problem, we design GENPass as follows.

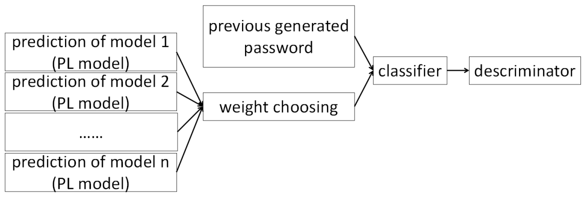
prediction of model 1 (PL model)
prediction of model 2 (PL model)
......
prediction of model n (PL model)
previous generated password
weight choosing
classifier
descriminator

Fig. 1. **The diagram of GENPass model.** The units are first generated by separate models. The weight choosing process selects one specific unit from these units. The determined unit by the weight choosing process together with the previous generated password will be fed into the classifier. The classifier will judge which model the input belongs to. The validity of the unit depends on the deviation judged by the descriminator.

1) Prediction of model n: All the models are the PL model mentioned in the previous section. We trained each model separately. Thus, given an input, the model can output the result with its own principle.

2) Weight Choosing: We assume that every model has the same probability, so the output of each model can be combined directly. The combined list is the input of the weight choosing process, and the output is a random choice. When an unit is determined, it will be replaced by a random alphabetic sequence. All the random selections follow the algorithm 1 mentioned in PL.

3) Classifier: The classifier is a CNN classifier [29] trained by passwords from different wordlists. Given a password, the classifier can tell which wordlist the password most likely comes from. Through a softmax layer, the output will be a list of numbers whose sum is one. For example, there are only two wordlists and the output of the classifier is (0.3, 0.7). This means the password is more likely to appear in the second wordlist.

4) Descriminator: The descriminator is used to judge whether the unit will be accepted. The unit generates a password string by weight choosing process. To generate a general wordlist, the descriminator should accept those passwords which seem plausible in every dataset. In another words, those passwords cannot be distinguished by the classifier. We use

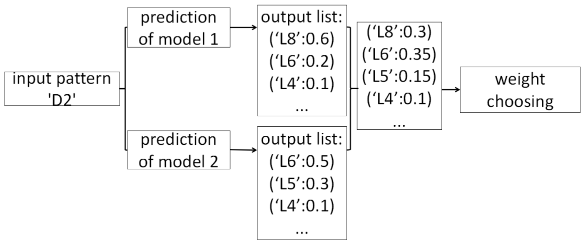
input pattern 'D2'
prediction of model 1
prediction of model 2
output list: ('L8':0.6) ('L6':0.2) ('L4':0.1) ...
output list: ('L6':0.5) ('L5':0.3) ('L4':0.1) ...
('L8':0.3) ('L6':0.35) ('L5':0.15) ('L4':0.1) ...
weight choosing

Fig. 2. **An example of weight choosing process.** For example, there are 2 models. The model 1's output is [(L8:0.6), (L6:0.2), (L4:0.1)...] and the model 2's output is [(L6:0.5), (L5:0.3), (L4:0.1)...]. (L8:0.6) means the unit is 'L8' and probability is 0.6 in that model. Because all the models have the same probability, the input to the weight choosing can be written as [(L8:0.3), (L6:0.35), (L5:0.15), (L4:0.1)...]. Therefore, the most possible unit is L6.

the standard deviation of the classifier's output to judge the password's generality. If the standard deviation is too large, the lately generated unit will be discarded because it is probably a special case in one dataset. Otherwise, the unit will be accepted. In our model, we choose 0.2 as a threshold of the standard deviation after several experiments. Since if the threshold is too small, the model will be much more time-consuming and the output will contains more duplicates.

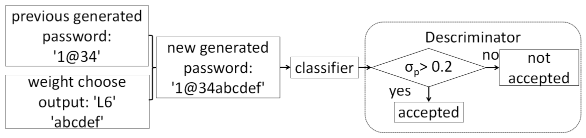
previous generated password: '1@34'
weight choose output: 'L6' 'abcdef'
new generated password: '1@34abcdef'
classifier
Descriminator
$\sigma_p > 0.2$
no
not accepted
yes
accepted

Fig. 3. **The detailed diagram of the classifier and the descriminator.** Assume that the new unit is 'L6'. The candidate password string chosen by weight choosing process is 'abcdef', and the previous generated password is '1@34', the input to the classifier will be '1@34abcdef'. If the output of the classifier is [0.4,0.6], the descriminator will accept 'L6' and 'abcdef' and print the password. If the output of the classifier is [0.3,0.7], 'L6' will be discarded since the standard deviation is over 0.2.

C. GENPass with probability

In the previous section, we assume that each dataset has the same probability. However, this assumption is unpractical. Some datasets are more typical so that passwords are more likely to come from these datasets. We introduce the concept of weight to solve this problem. Those typical datasets should have a high weight. The generality is judged by whether the weight of datasets corresponds with the distribution of the classifier's output. We improve the weight choosing process and descriminator in the previous model as follows. We also develop a method to calculate the weights in case the weights are not given.

1) Weight Choosing process with probability: We assume that a probability distribution P stands for each dataset's weight, where $\Sigma p = 1$. So the outputs of different models should be added according to weights. The weights are the probabilities. According to the Bayes Formula [30], the total probability of the output is still 1.

2) Descriminator with probability: The generality is judged by the distance between the weight of datasets P and the distribution of the classifier's output Q. Standard deviation

Algorithm 2: Calculate the distribution

```
1 Generate the distribution $P$ randomly;
2 repeat
3     generate 10 passwords by GENPass with probability
        without descriminator, denote the outputs of the
        classifier as $Q_i (i = 1...10)$;
4     calculate the average of classifier's outputs
        $Q = \frac{1}{10}\Sigma_{i=1}^{10} Q_i$;
5     Loss is the KL divergence of two distributions
        $Loss = \mathrm{D}_{KL}(P||Q)$;
6     Update the distribution $P$ by ascending its stochastic
        gradient times steps $P = P + \alpha \times \nabla_{Q-P} Loss$;
7     Make sure that the sum of P equals 1 by using
        $P_i' = \frac{P_i}{\Sigma P_i}$;
8 until the distribution is stable;
9 return distribution $P$;
```

cannot depict the distance between two distributions. So we introduce Kullback-Leibler divergence(KL divergence) [31] to depict it. The KL divergence is calculated as follows.

$$\mathrm{D}_{KL}(P||Q) = \Sigma p(x) \ln \frac{p(x)}{q(x)} \tag{1}$$

During the training process, we use gradient descent to make P approach Q. The distribution P is first generated randomly. In each iteration, ten new passwords are generated by GENPass with probability. These passwords are fed to the classifier and the results are recorded as $Q_i (i = 1...10)$. We denote the average of these results as the output of the classifier Q in order to eliminate random errors. We use KL divergence as the loss function. Updating the P by ascending $Q - P$ stochastic gradient times steps α makes P approach Q quickly and accurately. We assume steps $\alpha = 1$. We use $P_i' = \frac{P_i}{\Sigma P_i}$ to ensure the total probabilities is 1. We believe the distribution is stable when the loss is less than 0.001. The pseudo code is shown in Algorithm 2.

When P is determined after training, the criterion for accepting the password is whether the KL divergence is smaller than 0.1, a value we set.

IV. Experiment and Evaluation

In this section, we first depict our training and testing procedures. Then, we demonstrate the results of our experiments and analyze them. We used TensorFlow [32] version 1.0.0 with GPU support. All experiments were performed on a workstation running CentOS 7.0, with 192GB of RAM, an Intel(R) Xeon(R) CPU E5-2630 , and an NVIDIA GeForce GTX 1080 GPU.

A. Experimental data

Experimental data are all collected from leaked passwords from famous websites. Li et al. [24] has proved that passwords in different language environments have little generality. So in our experiments, we only choose English passwords. Our model is not designed to solve the generality between different languages. Our training and testing data are from Myspace [33], phpBB [34], RockYou [35] and LinkedIn [36]. [8] and [9] have proved that neural network model outperforms those traditional technologies such as HashCat [3] and JtR [4]. So we do not spend time on testing wordlists generated by HashCat and JtR. Detailed information is shown in Table II.

TABLE II
CONTRAST BETWEEN DIFFERENT DATESETS

Name	Language	Address	Number
MySpace	English	http://www.myspace.com/	37,144
phpBB	English	http://www.phpbb.com/	184,389
RockYou	English	http://www.rockyou.com/	14,344,391
LinkedIn	English	http://www.linkedin.com/	60,143,260

B. Training and Testing Methods

1) PL: To evaluate PL, we trained the model with Myspace [33] and phpBB [34] respectively. The testing datasets include Myspace, phpBB, RockYou [35] and LinkedIn [36]. After each training, the model generated a new wordlist. The tests can be divided into two types. One is called one-site test, in which the training and testing dataset are the same. The other is called cross-sites test, in which the training and testing datasets are different. Then, we enumerated the passwords in the generated wordlists to see if it was listed in the testing set and calculated the percentage. We randomly chose 10% passwords in the testing dataset as testing data. The final matching rate is the average of the results of the procedure above repeated for 10 times.

2) GENPass: To evaluate the GENPass model, we trained the model with Myspace and phpBB. The testing datasets include Myspace, phpBB, RockYou and LinkedIn. We calculated the matching rate by using the same method described in the previous experiments. Then we compare the results with those of the PL model trained by each wordlist individually. We also trained the PL model with a simple mixture of two wordlists. We compare the result with the one of GENPass model.

C. Evaluation

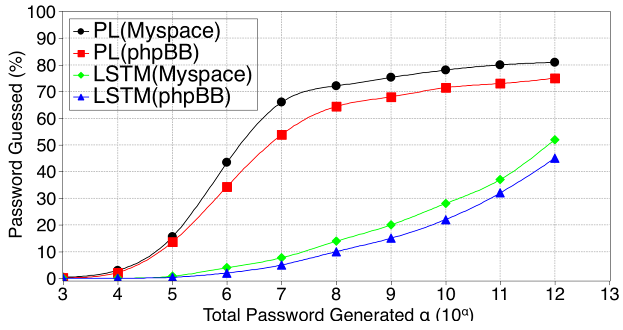

Fig. 4. **One-site tests of different models and different datasets.** PL and LSTM are models. Myspace and phpBB are training datasets.

1) PL: We respectively trained the PL and LSTM model with Myspace and phpBB and did the one-site test. The result is shown in Figure 4. For example, PL(Myspace) means the training model is PL and the training and testing datasets are both Myspace. It is obvious that the PL performs much better than LSTM. Although the matching rates of both models will finally reach a higher level with more guesses, PL can achieve the same matching rate with much less guesses compared with LSTM. For instance, when testing set is Myspace, LSTM guessed 10^{12} times to reach a 50% matching rate while PL only needed about 10^7 times. After generating 10^7 passwords, PL regenerates over 50% of the datasets while LSTM only regenerates less than 10%. The result proves that the PCFG rules help improve the efficiency of password guessing.

We also find that the model trained with Myspace performs better than that with phpBB. This phenomenon indicates that datasets have differences. The model trained by different datasets got different results. A suitable explanation is that the features in Myspace are easier to capture than in phpBB because the number of passwords in Myspace is smaller, as shown in Table II.

Table III shows the result of cross-sites tests at 10^6 and 10^7 guesses. We respectively trained the PL and LSTM model with Myspace and phpBB and used the outputs of these models to test RockYou and LinkedIn. The cross-sites tests always have worse performances than the one-site tests do. This indicates that different datasets have their own regularity. Learning from only one dataset is not enough to crack others. During the cross-sites tests, the PL model always has a better performance than the LSTM model with an increasement of 16%-30%. The result indicates that people are inclined to use some fixed strings, such as words and name, and this phenomenon is common in leaked passwords. PL is proved to be useful in cross-sites tests. However, the result is still unsatisfactory due to the lack of generality.

TABLE III
CROSS-SITES TEST RESULT

	Total Password Generate	Myspace-RockYou	phpBB-RockYou	Myspace-LinkedIn	phpBB-LinkedIn
PL	10^6	1.10%	0.41%	0.30%	0.10%
	10^7	3.47%	1.12%	1.08%	0.38%
LSTM	10^6	0.65%	0.37%	0.22%	0.07%
	10^7	2.80%	0.96%	0.83%	0.30%

2) GENPass: In all GENPass's tests, the training datasets are Myspace and phpBB. Table IV compares the GENPass model with the PL model whose training dataset is simply mixing Myspace and phpBB. The matching rate of GENPass is $40\% - 50\%$ higher than that of PL when the testing set is Myspace. However, they have a similar result when the testing set is phpBB. The PL model performs better when the size of the testing set is large. This result is probably attributed to following reasons. The larger dataset has a deeper influence on the output wordlist while the impact of the smaller one will be weakened. The smaller dataset can be regarded as a noise added to the larger one. For example, phpBB contains 106 '123456' out of 184389 passwords while Myspace contains 59 'password' out of 37144 passwords. The 'password' is more important than the '123456'. But in the simply mixing dataset, the 'password' plays a less important role than in the origin dataset. The simple mixture of two datasets, especially when two datasets have very different numbers of passwords, ruins the regularity of datasets. This conclusion proves the necessity of GENPass model. Our model overcomes these obstacles by choosing the output in two models respectively.

TABLE IV
DIFFERENCES BETWEEN GENPASS AND SIMPLY MIXING

Total Password Generated	GENPass -Myspace	GENPass -phpBB	PL(Simply Mixing) -Myspace	PL(Simply Mixing) -phpBB
10^6	31.58%	33.69%	19.22%	33.08%
10^7	55.57%	57.90%	38.87%	55.55%

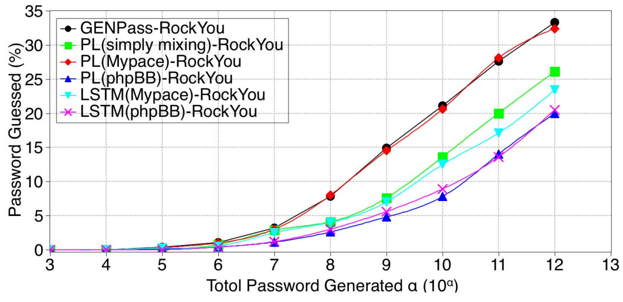

Fig. 5. **Cross-sites tests of different models and different datasets.** GENPass, PL and LSTM are models. Myspace and phpBB are training datasets. The target dataset is RockYou.

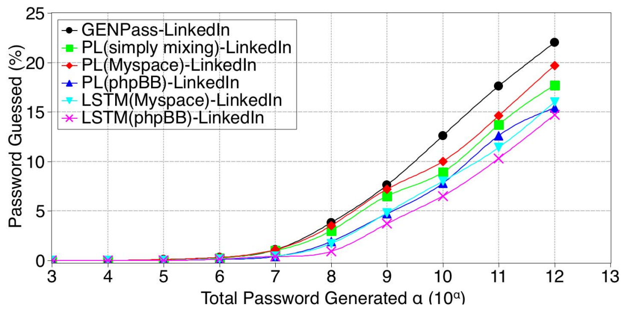

Fig. 6. **Cross-sites tests of different models and different datasets.** GENPass, PL and LSTM are models. Myspace and phpBB are training datasets. The target dataset is LinkedIn.

Fig 5 and 6 show the result of cross-sites tests when models are GENPass, PL and LSTM. GENPass uses Myspace and phpBB as training datasets, while the training datasets of PL and LSTM is specified in the parentheses. The testing set of Fig 5 and Fig 6 are RockYou and LinkedIn respectively.

Taking both Figure 5 and 6 into consideration, the GENPass model outperforms all the other models. Using raw LSTM without any preprocessing performs worst in two tests. Using PL to train Myspace alone performs second best. This proves that Myspace is a good dataset. The passwords in Myspace are typical among other datasets. Simply mixing two datasets cannot improve the matching rate. Instead, the matching rate will drop because of the bad dataset, namely phpBB in this test. The result proves that GENPass can overcome the discrepancies between datasets and achieve a higher matching rate.

V. Conclusion

This paper introduces GENPass, a general password guessing model. GENPass borrows the idea from PCFG [6] and GAN [28] in order to improve the matching rate of the generated password lists. Our results show that word-level (if tags, such as "L4" "D3" "S2", are viewed as words) training outperforms character-level training [8], [9]. Furthermore, we implement adversarial generation from several datasets and the result shows that multi-source generation achieves a higher matching rate when doing cross-sites test.

Our work is greatly inspired by previous researches, especially by the excellent job of Melicher et al. [8]. Before then, the researches of password guessing mainly focused on the innovations in Markov model and PCFG. Melicher et al. first proved that neural network had a better performance than the probabilistic method. Neural network is believed to have the ability to detect the relationship between characters that probabilistic methods do not. In this paper, we first extend the character-level training to word-level training by replacing letters, digits and special chars with tags, namely, PCFG+LSTM(PL). In one-site tests, the PL model generated 10^7 passwords to achieve a 50% matching rate while the LSTM model must generate 10^{12} passwords. In cross-sites tests, PL increased the matching rate by $16\%-30\%$ compared with LSTM when learning from a single dataset. Our model also extends single dateset training to multi-dataset training and uses adversarial generation to ensure the output passwords do not contain features in a certain dataset, which means the output is general to all. The matching rate of GENPass is 20% higher than that of simply mixing several datasets at the guess number of 10^{12}. Limited by the number of leaked passwords, we use Myspace and phpBB as training data, RockYou and LinkedIn as testing data. But the result is enough to prove GENPass is effective in this task.

We believe password guessing deserves further research as password authentication will not be totally replaced for a long time. However, the scarcity of training resources remains a problem. Because of the security protocols, we have limited access to passwords, especially those we need for training and testing. To solve this problem, studies on transfer learning may help.

VI. Acknowledgements

National Natural Science Foundation of China (61571290,61431008), NSF grants 1652669 and 1539047.

References

[1] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*. Nature 521.7553 (2015): 436-444.

[2] Silver D, Huang A, Maddison C J, et al., *Mastering the game of Go with deep neural networks and tree search*. Nature, 2016, 529(7587):484.

[3] STEUBE, J. Hashcat. https://hashcat.net/oclhashcat/.

[4] PESLYAK, A. John the Ripper. http://www.openwall.com/ john/..

[5] Ma J, Yang W, Luo M, et al., *A Study of Probabilistic Password Models*. Security and Privacy. IEEE, 2014:689-704.

[6] Weir M, Aggarwal S, Medeiros B D, et al., *Password Cracking Using Probabilistic Context-Free Grammars*. Security and Privacy, 2009, IEEE Symposium on. IEEE, 2009:391-405.

[7] Lipton Z C, Berkowitz J, Elkan C., *A critical review of recurrent neural networks for sequence learning*. Computer Science, 2015.

[8] Melicher W, Ur B, Segreti S M, et al., *Fast, lean and accurate: Modeling password guessability using neural networks*. Proceedings of USENIX Security. 2016.

[9] Hitaj B, Gasti P, Ateniese G, et al., *PassGAN: A Deep Learning Approach for Password Guessing*. arXiv preprint arXiv:1709.00440, 2017

[10] Yi P, Zhu T, Zhang Q, Wu Y, Pan L, *Puppet Attack: A Denial of Service Attack in Advanced Metering Infrastructure Network*. Journal of Network and Computer Applications, 2016, 59(1):325-332

[11] Zhu T, Yu M. *A Dynamic Secure QoS Routing Protocol for Wireless Ad Hoc Networks*. IEEE Sarnoff 2006, Princeton, NJ, April 2006

[12] Yi P, Zhu T, Ma J, Wu Y, *An Intrusion Prevention Mechanism in Mobile Ad Hoc Networks*. Ad Hoc & Sensor Wireless Networks, 2013, 17(3):269-292

[13] Wang X, Yi P, *Security Framework for Wireless Communications in Smart Distribution Grid*. IEEE Transactions on Smart Grid, 2011, 2(4):809-818

[14] Yi P, Zhu T, Zhang Q, Wu Y, Li J, *A Denial of Service Attack in Advanced Metering Infrastructure Network*. ICC2014, Australia, 2014

[15] Yi P, Zhu T, Zhang Q, Wu Y, Li J, *Green Firewall: an Energy-Efficient Intrusion Prevention Mechanism in Wireless Sensor Network*. GLOBECOM 2012, USA, December, 2012

[16] Li Y, Zhu T. *Gait-Based Wi-Fi Signatures for Privacy-Preserving*. In the 2016 ACM Symposium on InformAtion, Computer, and Communications Security, Xi'an, China, April, 2016

[17] Yi P, Zhu T, Liu N, Wu Y, Li J, *Cross-layer Detection for Black Hole Attack in Wireless Network*. Journal of Computational Information Systems, 2012, 8(10):4101-4109

[18] Zhu T, Xiao S, Yi P, Don Towsley, Gong W, *A Secure Energy Routing Mechanism for Sharing Renewable Energy in Smart Microgrid*.IEEE SmartGridComm2011, Brussels, Belgium, 17-20 October 2011

[19] Yi P, Wu Y, Liu N, Wang Z, *Intrusion Detection for Wireless Mesh Networks using Finite State Machine*. China Communications, 2010, 7(5):40-48

[20] Yi P, Wu Y, Zou F, Liu N, *A Survey on Security in Wireless Mesh Networks*. IETE TECHNICAL REVIEW, 2010,27(1):6-14

[21] Zhu T, Yu M. *A Secure Quality of Service Routing Protocol for Wireless Ad Hoc Networks*. IEEE GLOBECOM 2006, San Francisco, USA, November 2006.

[22] Yi P, Jiang X, Wu Y, *Distributed Intrusion Detection for mobile ad hoc networks*. Journal of Systems Engineering and Electronics, 2008,19(3):851-859

[23] Narayanan A, Shmatikov V., *Fast dictionary attacks on passwords using time-space tradeoff*. ACM Conference on Computer and Communications Security. 2005:364-372.

[24] Li Z, Han W, Xu W, *A Large-Scale Empirical Analysis of Chinese Web Passwords*. Usenix Security. 2014: 559-574.

[25] Wang D, Zhang Z, Wang P, et al., *Targeted Online Password Guessing: An Underestimated Threat*. ACM Sigsac Conference on Computer and Communications Security. 2016:1242-1254.

[26] Li Y, Wang H, Sun K., *A study of personal information in human-chosen passwords and its security implications*. Computer Communications, IEEE INFOCOM 2016

[27] Graves A., *Generating Sequences With Recurrent Neural Networks*, Computer Science, 2013.

[28] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. *Generative adversarial nets*. International Conference on Neural Information Processing Systems. MIT Press, 2014:2672-2680.

[29] Zhang, Xiang, Junbo Zhao, and Yann LeCun. *Character-level convolutional networks for text classification*. Advances in neural information processing systems. 2015.

[30] Stuart, A.; Ord, K. (1994). *Kendall's Advanced Theory of Statistics: Volume IDistribution Theory*, Edward Arnold, 8.7.

[31] MacKay, David J.C. (2003). *Information Theory, Inference, and Learning Algorithms* (First ed.). Cambridge University Press. p.34.

[32] TensorFlow. https://www.tensorflow.org

[33] Myspace. http://www.myspace.com/

[34] phpBB. http://www.phpbb.com/

[35] RockYou. http://www.rockyou.com/

[36] LinkedIn. http://www.linkedin.com/