TG-SPSR: A Systematic Targeted Password Attacking Model

Mengli Zhang[1], Qihui Zhang[1], Wenfen Liu[2*], Xuexian Hu[1]and Jianghong Wei[1]
[1] State Key Laboratory of Mathematical Engineering and Advanced Computing
Henan 450001, China
[e-mail: zml1122y@163.com]
[2] School of Computer Science and Information Security, Guangxi Key Laboratory of Cryptogpraphy and Information Security, Guilin University of Electronic Technology
Guangxi 541004, China
[e-mail: liuwenfen@guet.edu.cn]
*Corresponding author: Wenfen Liu

Received July 20, 2018; revised October 25, 2018; accepted November 21, 2018; published May 31, 2019

Abstract

Identity authentication is a crucial line of defense for network security, and passwords are still the mainstream of identity authentication. So far trawling password attacking has been extensively studied, but the research related with personal information is always sporadic. Probabilistic context-free grammar (PCFG) and Markov chain-based models perform greatly well in trawling guessing. In this paper we propose a systematic targeted attacking model based on structure partition and string reorganization by migrating the above two models to targeted attacking, denoted as TG-SPSR. In structure partition phase, besides dividing passwords to basic structure similar to PCFG, we additionally define a trajectory-based keyboard pattern in the basic grammar and introduce index bits to accurately characterize the position of special characters. Moreover, we also construct a BiLSTM recurrent neural network classifier to characterize the behavior of password reuse and modification after defining nine kinds of modification rules. Extensive experimental results indicate that in online attacking, TG-SPSR outperforms traditional trawling attacking algorithms by average about 275%, and respectively outperforms its foremost counterparts, Personal-PCFG, TarGuess-I, by about 70% and 19%; In offline attacking, TG-SPSR outperforms traditional trawling attacking algorithms by average about 90%, outperforms Personal-PCFG and TarGuess-I by 85% and 30%, respectively.

Keywords: password authentication, targeted password attacking, BiLSTM, probabilistic context-free grammar, Markov chain, personal information

This work is supported by the National Natural Science Foundation of China under Grant No. 61502527, No. 61702549, No. 61862011, No. 61872449 and Guangxi Natural Science Foundation under Grant No. 2018GXNSFAA138116 and the Guangxi Key Laboratory of Cryptography and Information Security under Grant No. GCIS201704.

1. Introduction

Passwords have become one of the most widely used technologies in the field of identity authentication because of its simple, flexible and easy-to-use features [1]. Passwords will also exist as the most important authentication method for a long time [2-4]. With the increasing number of registered websites, users inevitably tend to choose passwords with low information entropy so as to keep them in mind [5]. This fragile password behavior is easily exploited by illegal attackers to mount an effective attack. In order to enhance the security of passwords, researchers simulated a series of cracking algorithms to guide the user's policy to construct passwords [6-11]. Among them, Weir et al.. [7] and Nayanan et al.. [6] proposed probability password cracking models: probabilistic context-free grammar (PCFG) and Markov chain-based algorithms, which made notable contributions to the study of password cracking algorithms. Furthermore, other researchers have conducted a series of researches on the basis of these two algorithms [8].

However, with the continuous occurrence of large-scale personal information leakage incidents, various types of personally identifiable information (PII) and passwords used by users on other websites are increasingly accessible to attackers [12-14]. Attackers will obtain huge economic benefits by accurately attacking the victim's bank account, e-mail, etc. Driven by profit, a large number of criminals start attacking certain targets. Targeted online attacking become a very serious and realistic threat [11,15-17]. For example, in April 2016, a citizen's information disclosure incident in Turkey involved humanity accounting for 64% of Turkey's total population, totaling 50 million Turkish citizens [13]. According to a report by CNNIC (China interNet Network Information Center) in 2015, more than 78.2% of the 668 million Chinese netizens had experienced personal information data leakage [18]; In recent years, a series of recent information breaches in the United States have caused more than 253 million U.S. Internet users to become victims of personally identifiable information(PII) and password leaks [19].

This implies that existing password generation rules built on trawling attacking probabilistic models only consider the limited off-line guessing threat [20,21], but cannot defend more and more realistic and increasingly harmful targeted online guessing attacking [11]. Therefore, there is an urgent need to comprehensively evaluate the application of personal information in user passwords, password reuse and modification modes, and quantify security risks. A complete password attacking framework is a prerequisite. However, current research on targeted password attacking is far from being a system, and sporadic research results are not sufficient to lead the industry [22].

In particular, there are many challenges in how to make full use of available information for targeted attacking. First, the type and structure of private information are extremely different. Some kind of PII (e.g. name, and hobby) are consist of letters, some (birthday and phone number) consist of digits, and some (e.g., user name) are a mixture of letters, digits and symbols. Thus, how to establish an attack model to automatically identify highly heterogeneous PII in passwords is a crucial problem. Second, users select a variety of modification rules to modify passwords for cross-site reuse, such as insertion, deletion, capitalization and leet (e.g. $password \rightarrow p@ssword$) and the hybrid ones (e.g., $password \rightarrow p@ssword123$), that users can utilize to create analogous passwords. It is quite arduous to characterize these rules systematically and legitimately allocate them for each independent user. Third, existing probabilistic password cracking models either focus on structure-level mining or character-level mining. Although they have achieved good results, they all have advantages and disadvantages. It isn't easy to build a fusion model to overcome their shortcomings on the basis of existing models.

1.1 Related Works

Narayanan and Shmatikov [6] migrated the Markov chain technology from natural language processing to password attacking for the first time, and put forward a Markov password attacking model based on n-gram. Although the method still has some limitations (the password is limited to a certain probability threshold, and only the first-order and second-order Markov chains are used), the heuristic idea is removed and the password probability model is established, which provided new ideas for later research. In 2009, Weir et al.. [7] proposed a structure-based password attack model, probabilistic context-free grammar (PCFG), that learns the structure distribution of the password from the training set, and describes this distribution in the form of probability. Although this method may have an overfitting phenomenon when the training set is too large, it can well characterize the structure of the password. The two passwords attacking algorithms have different advantages: the Markov chain-based model can adjust the generalization ability by changing the order, and the PCFG-based model can accurately abstract the basic password structure.

Zhang et al.. [15] conducted a study on large-scale password data containing 7 million pairs of passwords (old password, modified password). They proposed a heuristic prediction algorithm to forecast the new password of the same account from the user's old password on an account. In 2013, Castelluccia et al.. [23] also explored personal information used in passwords, such as email, birthday and username, but did not take into account previous passwords or cross-site passwords. In 2014, Das et al.. [16] studied the threat posed by user password reuse behavior and proposed a cross-site password guessing algorithm. However, there are still some improvements in their algorithms for password reuse and modification: First, manually assigning modification rules, all users are assigned rules according to a fixed priority, but in fact, password modification and reuse vary with each individual, and that there is a close context with the original password. Second, not taking into account the combination rules. Third, Heuristic-based.

Li et al.. [17] analyzed the usage of name, birthday, email, mobile phone number and user names in user passwords on the 12306 (Chinese ticketing website) dataset, and proposed a PCFG-based targeted password attacking algorithm, Personal-PCFG. A new grammar variable (B: birthday, N: name, E: E-mail, P: phone number, A: account) was added for each type of private information in the basic structure for passwords, with subscript to indicate the length. For instance, **Zhang San** born on June 5, 1980 has a password ***zhang8065*** that will be parsed into N_5B_4 during the training phase. In the same year, wang et al.. [11] found that Li et al..'s length-based private information is not accurate, because the length is random and can lead to ambiguous probability. For example, in an extreme case, users in the training set al.l use family name to construct passwords, the length is 3, then only his given name would be used when attacking **Zhang San**. This is obviously unreasonable. Therefore, they proposed type-based PII tags, where each type of personal information is represented by a definitive label rather than length. For instance, N1 for the usage of full name, N2 for the abbr. of full name (e.g., ***sz*** from ***San Zhang***), B1 for full birthday in YMD format (e.g., ***19800605***), B2 for full birthday in MDY, and so on.

1.2 Our Contributions

Note that, existing methods for targeted password attacking are still fragmented and nonhierarchical, either using PII or password reuse and modification alone, and do not maximally integrate all the information related to the attacked target in an independent model. In summary, we make the following key contributions in this work:

- **A targeted password attacking model**: We synthesize the advantages of PCFG and Markov chain models, and propose a targeted password attacking model based on structure partition and string reorganization, which is denoted as TG-SPSR: the basic structure of password is divided into different segments using PCFG algorithm. At the same time, the Markov model is used to model the strings in each segment to generate new strings to achieve both accuracy and generalization ability of the model. Our model mines the password features from structural level and character level, which is more consistent with the human habit of constructing passwords.
- **Three new structural grammars**: In the basic structure of the password, we insert the type-based PII pattern, the trajectory-based keyboard pattern on the basis of the original grammars, and distinguish the position of special characters in the password. Our structural grammars make a more granular division of the password structure, resulting in a more accurate probability distribution.
- **A password modification framework**: Based on the BiLSTM recurrent neural network, we construct a password modification rule classifier: starting from the structure and character characteristics of the input password, the classifier can determine which rules the user will use to modify the password. We collated two datasets from large-scale online data through E-mail matching, CEm-dataset and EEm-dataset. There are approximately 4 million records in Chinese dataset and about 6 million records in English dataset. Each record is commensurate with a user, containing a E-mail and two passwords from different websites. For the classification problem of Chinese users, we use CEm-dataset to train the model, EEm-dataset for English users, which free us from heuristic thinking and handle each password automatically.

2. Targeted Attacking Model

In this section, we propose a targeted password attacking algorithm based on structure partitioning and string reorganization. We present our overall model. The details of the stand-alone module will be elaborated in sections 4, 5, and 6.

3.1 Overall Framework

The systematic targeted password attacking model in this paper is mainly composed of three basic modules, structure partition module, string reorganization module, and rule-based password modification module: (1) The structure partition module mainly excavate the user's habit for constructing passwords from the structural level. The basic structure sets and the string sets are obtained by dividing the password in the training set. (2) The string reorganization module starts from the character level, using the Markov chain-based model to reorganize the strings sets that obtained from the structure partitioning module to generate new string sets. Letters and digits make up the majority in password. So using Markov chain-based model in the L and D segments can make full use of the training set to extract the user's behavioral characteristics for constructing passwords. (3) The principal function of rule-based password modification module is to classify the generated passwords and get the probability that the password belongs to each class using the trained BiLSTM recurrent neural network classifier. If the victim is a Chinese user, then we use the CEM-dataset to train the classifier, EEm-dataset for English users. Ultimately, the password is modified according to different modification rules to generate new passwords sorted in descending probability order to obtain an extended targeted password attacking dictionary. The model is illustrated in **Fig. 1**.

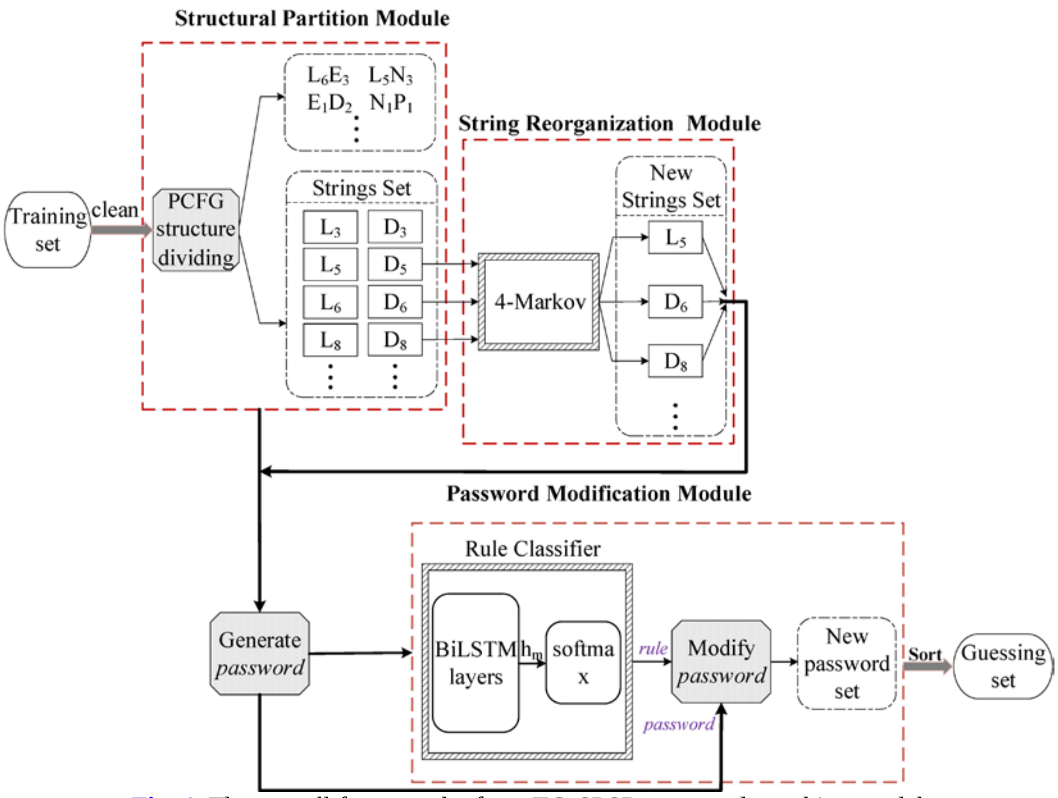


Fig. 1. The overall framework of our TG-SPSR password attacking model

3.2 Algorithm Description

The model of this paper is divided into two stages of training and guessing set generation.

Training phase: (1) Presetting initial grammar: letter segment (L), digit segment (D), special character segment (S), and 5 PII segments, A(Account), B(Birthday), E(E-mail), N(Name), P(Phone Number), establishing a probability model based on the PCFG structure partition algorithm [7]. After training, we can obtain the basic structure set (*BaseStructSet*) and string sets (*StringSet*); (2) Using the Markov chain-based model [6] to perform string reorganization on substrings for L and D segments of the same length and length greater than or equal to 4 in the *StringSet* to generate a new substring set (*NewTypestring*). Afterwards, using *NewTypestring* to replace the string with the same structure in the *Stringset*. Ultimately, getting a new string set (*M-StringSet*).

Guessing set generation phase: Based on the *BaseStructSet* and *M-StringSet*, using the "NEXT" algorithm in [7] to generate the candidate passwords. Each of them is classified by the BiLSTM recurrent neural network classifier to get the probability distribution of classification rules, then the modified passwords are sorted in decreasing order of probability to generate the targeted password attacking dictionary. The detailed algorithm flow is presented in **Algorithm 1**.

In **Algorithm 1**, the grammar of the basic structure can be added and changed without affecting the overall algorithm. If we dig out more rich password structure features, it will be effortlessly to insert the grammar for verification.

Algorithm 1. Targeted Password Generation Algorithm Based on Structure Partition and String Reorganization.

```
input: TrainingSet; The size of the dictionary (n); Hidden Layers Number (H); Neurons Number per Layer(N)
output: GuessingSet
1.  BaseStructSet, StringSet ← Structure Partitioning(TrainingSet)
2.  FOR each TypeString IN StringSet
3.      IF TypeString == L OR D and len (TypeString) > 4
4.          NewTypeString ← 4-Markov(TypeString)
5.          M-StrintSet.push (NewTypeString)
6.      ELSE
7.          M-StrintSet.push(TypeString)
8.      END IF
9.   END FOR
10.  IF len(PasswordSet) < n OR BaseStructSet is not Null
11.      password ← generate password
12.     Rule vector ← BiLSTM classifier(password)
13.     FOR rule IN Rule vector
14.         Newpasswords ← modify(password, rule)
15.        GuessingSet.push (Newpasswords)
```

4. Structure Partition Module

In this section, we analyze the structural characteristics of the password in detail. We deal with structure partition similar to the original PCFG-based probabilistic model proposed by Weir et al.. [7] and consider users' personally identifiy information (PII) as type-based PII tags as [11]. Additionally, we integrately consider the influence of position distribution of special characters and common keyboard patterns by introducing the index bits to distinguish the position of special characters in the password and define a trajectory-based keyboard pattern in the basic grammar.

4.1 Defining Personal Information

In the structure partitioning module, we define the type of the string. The password structure is partitioned according to different types during the training process. After the training, we obtain the passwords structural distribution rules of the training set in the form of probability. At the same time, we store the string by type as a training set of the structure reorganization module.

We add 6 type-based PII to basic structure grammar as [11], as follows: (1) Account (A), A_1 for full account name (e.g., ***lemon123***), A_2 for the (first) letter-segment of account name (e.g., ***lemon***), A_3 for the (first) digital segment of account name (e.g., ***123***); (2) Birthday (B), B_1 for Year-Month-Date (e.g., ***19800605***), B_2 for Month-Date-Year, B_3 for Date-Month-Year (e.g., ***06051980***), B_4 for the date, B_5 for the year, B_6 for Year-Month (e.g., 1***98006***), B_7 for Month-Year (e.g., ***061980***), B_8 for the last two digits of year followed by Month-Date (e.g., ***800605***), B_9 for last two digits of year (e.g., ***060580***), B_{10} for Date-Month followed by the last two digits of year (e.g., ***050680***); (3) E-mail (E), E_1 for the full email prefix (e.g., ***sunshine520*** from ***sunshine520@example.com***), E_2 for the first letter segment of email prefix (e.g., ***sunshine***), E_3 for the first digital segment of account name (e.g., ***520***); (4) Name (N), N_1 for full name, N_2 for the abbr. of full name (e.g., ***sz*** from ***San Zhang***), N_3 for last name (e.g., ***zhang***), N_4 for first name, N_5 for the 1st letter of the first name followed by last name (e.g., ***szhang***), N6 for last name followed by the 1st letter of the first name (e.g., ***zhangs***), N7 for family name with its 1st letter capitalized (e.g., ***Zhang***); (5) Phone Number (P), P_1 for the full number, P_2 for the first three digits, P3 for last four digits.

In the training phase, we divide the passwords into letter segments (L), digit segments (D), special strings segments (S) and PII segments by character type, assuming that different types of segments are independent on each other. For example, in the training phase "zhang123#" is represented as $N_3D_3S_1$ that is defined as the basic structure of the password. At the same time, we can obtain the string sets such as N_3 :Zhang, D_3:123, S_1:#. In the password generation phase, the basic structure is matched by searching the string sets to generate candidate passwords that are sorted in descending order of probability. For example, the probability of password "zhang123#" is: $P(zhang123\#) = P(N_3D_3S_1) \cdot P(N_3 \to zhang) \cdot P(D_3 \to 123) \cdot P(S_1 \to \#)$.

4.2 Distribution of Special Characters

In general, inserting special characters to the password is a common way for people to increase the password strength. When the previous PCFG-based models divide special characters, the default assumption was that the position of the special characters in the password was evenly distributed, thus didn't distinguish special characters. However, our statistical results show that the distribution of special characters is closely related to their location. Our evaluation builds on seven large-scale real-world password datasets (see **Table 1**), including half from the Chinese websites and the other from English websites that cover many mainstream web services. The datasets in **Table 1** are publicly available for research on the internet. Some of them have been widely used for trawling password models.

Table 1. Basic info about the datasets without personal info

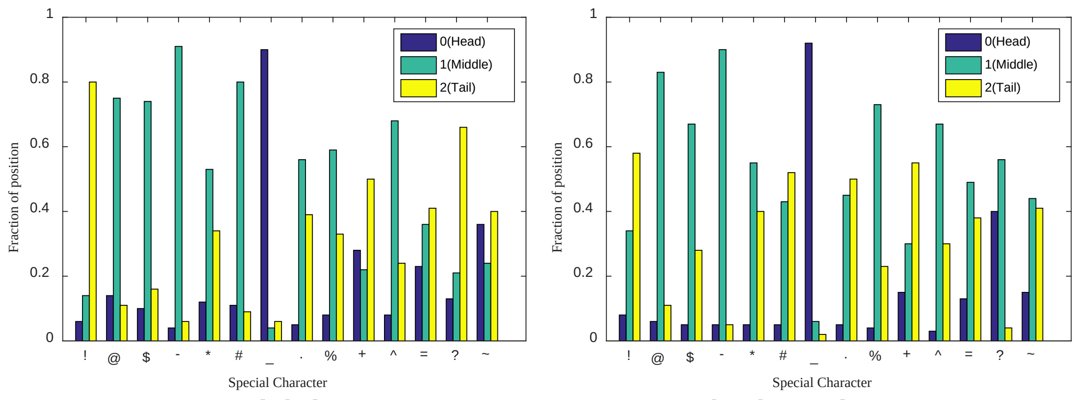
Dataset	Web service	Language	Leaked Time	Valid Account
CSDN	Programmer	Chinese	Dec., 2011	6,419,288
Taobao	E-commerce	Chinese	Dec., 2011	25,698,453
7k7k	Mini-games	Chinese	Nov., 2011	24,156,142
Rockyou	Social forum	English	Dec., 2009	32,497,566
000webhost	Web hosting	English	Oct., 2015	15,197,451
Twitter	Social media	English	Nov., 2016	35,256,881

Table 2 shows the distribution of commonly used special characters in our datasets, where 0 means that it appears at the head of the password; 1 stand for the middle of the password; 2 for the tail of the password. From the **Table 2** we can see that the distribution of special characters is extremely uneven, the special characters are less distributed at the head of the password (e.g., **?**), some characters appear more frequently at the tail of the password (e.g., **?**), and some frequently appear in the middle (e.g., **@**). Further, we summarize in **Fig. 2** where 14 commonly used special characters often appear in passwords, finding that 57 % of them often appear in the middle, 36 % for tail and just 7% for head.

Notice that, the probability that the traditional PCFG-based algorithms count a special character during the training phase is the sum of the probabilities for all positions (assuming uniform distribution), which is inaccurate. Therefore, our model adds an index bit to distinguish their position information during the structure partition phase, consistent with our settings in **Table 2**.

Table 2. Distribution of special characters

Type	Index	Rockyou	000webhost	Twitter	CSDN	Taobao	7k7k
!	0	0.07	0.05	0.07	0.17	0.08	0.07
	1	0.17	0.09	0.12	0.36	0.37	0.32
	2	0.76	0.86	0.81	0.57	0.55	0.61
@	0	0.13	0.15	0.16	0.08	0.08	0.03
	1	0.75	0.75	0.73	0.68	0.78	0.89
	2	0.12	0.10	0.11	0.24	0.14	0.08
–	0	0.92	0.91	0.90	0.93	0.89	0.91
	1	0.06	0.02	0.02	0.02	0.04	0.06
	2	0.02	0.07	0.08	0.05	0.07	0.03



(a) English datasets (b) Chinese datasets

Fig. 2. Distribution of special characters

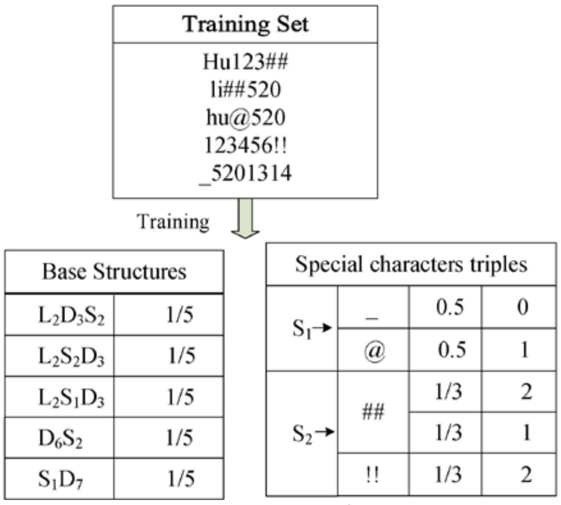
During the training phase, special characters of the same length but different positions will be counted separately and treated as different basic structure segments, represented by the triple (segment type, position, number). For example, in the training set, the passwords ***Hu123##*** and ***li##520*** both contain the string *##*, but since they are in different locations, they cannot be represented in general by S2, but instead generate two triples respectively, (S2, 2, 1) and (S2, 1, 1). During the string generation phase, based on the position of special characters in different basic structures, the corresponding triples are selected to calculate the probability. The index bits make finer partition of special characters and can more fully tap the user's habit of constructing passwords. The processing of special characters is shown in **Fig. 3**. Among them, strings with S2 tags are *##* and ***!!***, and the counts are 2 and 1, respectively. However, since *##* appears in two different positions, the middle and the tail. So not calculate the probability: $p(\#\#)=2/3$, $p(!!)=1/3$, but: $p(\#\#,1)=1/3$, $p(\#\#,2)=1/3$, $p(!!,1)=1/3$. For example, ***li123##***, when the special character position is not distinguished, the probability is as follows:

$$p(li123\#\#) = p(\mathrm{S} \to \mathrm{L}_2\mathrm{D}_3\mathrm{S}_2) \cdot p(\mathrm{L}_2 \to li) \cdot p(\mathrm{D}_3 \to 123) \cdot p(S_2 \to \#\#)$$
$$= \frac{1}{5} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{2}{135} \tag{1}$$

After increasing the index, the probability is as follows:

$$p(li123\#\#) = p(\mathrm{S} \to \mathrm{L}_2\mathrm{D}_3\mathrm{S}_2) \cdot p(\mathrm{L}_2 \to li) \cdot p(\mathrm{D}_3 \to 123) \cdot p(S_2 \to (\#\#, 2))$$
$$= \frac{1}{5} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{135} \tag{2}$$

Our model searches the string exactly according to the type of segments during the password generation phase, so integrates the index bit has negligible impact on the algorithm's time complexity. In addition, the introduction of index bits increases the storage space of the tables, but the proportion of special characters in the password is very small, so it doesn't affect the efficiency of the algorithm.



Base Structures	
$L_2D_3S_2$	1/5
$L_2S_2D_3$	1/5
$L_2S_1D_3$	1/5
D_6S_2	1/5
S_1D_7	1/5

Special characters triples			
$S_1 \to$	_	0.5	0
	@	0.5	1
$S_2 \to$	##	1/3	2
		1/3	1
	!!	1/3	2

Fig. 3. The training process of special character

4.3 Keyboard Pattern

Shiva et al.. [10] added the keyboard pattern to the basic structure of the password, achieving better effect than PCFG. But their definition of keyboard pattern is very broad. They only considered the physical positional relationship of each key without taking into consideration people's subjective consciousness. But in fact, humans usually follow a certain trajectory and order when constructing passwords using the keyboard pattern. Therefore, we propose a new keyboard pattern based on the user's keystroke trajectory.

In this subsection, we discuss how to define keyboard patterns and identify them in the passwords. Using the consequence to update the basic structure of the password after incorporating this pattern into probabilistic context-free grammar. A keyboard pattern is generally viewed as a sequence of keystrokes on the keyboard without regards to the actual characters typed. The sequence following a certain trajectory or graph can be easily remembered. One of the most common examples is ***asdfgh***, that starts with the letter ***a*** followed by the next 5 letters in sequence to the same row. Keyboard pattern is only a part of a password that frequently combined with other ingredients, for example ***asdfgh123***.

4.3.1 Defining Trajectory-Based Keyboard Patterns

Generally, users tend to construct their password following certain keyboard rules, which generate a fixed trajectory or graph of typing. For instance, most users used to pressing the keys on the same rows or columns, others even press the keys of a certain symmetrical area at the same time. In order to simulate the habit of constructing password for users, we define the keyboard patterns based on three kinds of trajectories in the basic structure of the password, and introduce the keyboard pattern into the grammar as a new non-terminal symbol K. As follows:

1. The keys in the same row, row keyboard pattern. We classified the combined keys contained at least 3 keys as a keyboard pattern, each key (except the first one) in them is physically next to the previous one (right or left) and keeping the same direction. For example, ***qwertyui*** $\rightarrow K_7$, the keyboard pattern starts with ***q*** and each key in them followed by the right key.

2. The keys in the same column, column keyboard pattern. The trajectory is the same as the row keyboard except that the direction transformed into up and down. For instance, ***1qaz*** $\rightarrow K_4$. In particular, The maximum length of a column keyboard string is 4 due to the structure of the keyboard.

3. Symmetrical block keys, block keyboard pattern. In the block keyboard, we define a basic block structure containing 4 adjacent keys, two up and down, such as ***qwas***. New block structures are formed based on the basic block structure by adding the same number of keys in each row or each column. When parsing a block structure, we either conduct only by rows or columns, but the direction in the same row or column must be consistent, in different rows and columns it can be inconsistent. For example, ***qweasd*** $\rightarrow K_6$.

4.3.2 Identifying Keyboard Patterns

Obviously, how to distinguish substrings from keyboard pattern or other patterns (L, D, S, etc) in passwords becomes both an emphasis and a nodus. It should be clear that any keyboard structure also has an original base structure (without using K). When dividing the password structure, the keyboard pattern should have higher priority than the original basic pattern.

In our mind, the keyboard pattern is to simulate the user's percussion trajectory on the keyboard. So when recognizing the keyboard pattern we should have a unified direction because the fixed typing order is more convenient for the user to remember, which means that there are no duplicate keys in single substring of keyboard pattern. This is different from the keyboard pattern mentioned in [10], who only consider whether adjacent characters in the substring are adjacent (or duplicated) on the keyboard. A typical example is ***qwwq***, that is thus K_4 in [10] but not keyboard pattern by us.

Notice that, since L segment, D segment, S segment are independent, the grammars in PCFG are unambiguous. By introducing K, we will face equivocal problem because a single K segment can contain all types of characters. Therefore, the K segment in the structure partition phase has a higher priority, that is, the substrings first match the keyboard pattern, and it wouldn't match L, D, S, if it belong to K. But all segments have an equal status in the strings generation stage. When dividing the password structure, if the substrings satisfy both the block keyboard and row keyboard or column keyboard key, we will first match the block keyboard that is of maximal length. For example ***qwedsazxc*** is classified as K_9 not $K_3K_3K_3$.

4.3.3 Probability Smoothing

We analyze the keyboard pattern and introduce smoothing during the training phase in this section. The smoothing technique is to eliminate overfitting problems in the dataset [8]. The probabilistic password cracking system will generate a string that does not appear in the dataset to generate a password with smoothing.

In order to smooth, we insert three transitional patterns according to the trajectory during training. Among them, R, C, B, respectively represent: row keyboard pattern, column keyboard pattern, block keyboard pattern. After counting the probability of all the keyboard pattern, we classify them by the transitional patterns. Without smoothing, only the items in the complete list for K_4 and K_6 patterns found during the training phase.

In particular, we choose Laplace smoothing to conduct keyboard patterns. The basic Laplace smoothing as shown in Equ. 3 that：

$$prob(i) = (N_i + \delta) \big/ (\sum N_i + C * \delta) \tag{3}$$

For the keyboard pattern, we regard a smoothed object as a specific transitional pattern applied to the direction of trajectory and practicable starting character. The probability that a keyboard pattern is smoothed is shown in Equ. 4 that：

$$prob(p) = prob(T) \frac{N_i + \delta}{\sum N_i + C * \delta} \tag{4}$$

where $prob(T)$ is the probability of the transitional pattern *T*, N_i is the counting of the ith keyboard pattern of this transitional pattern found in the training set, δ is a smoothing value in this article is 0.1, $\sum N_i$ represents the sum of keyboard patterns found in *T*, *C* is the total number of unique keyboard patterns for *T*.

In this approach, if we find a keyboard pattern ***asdf*** in the training set with a transitional pattern of R_4 and direction from left to right. In the smoothing, we change the start character ***a*** to ***q*** or other characters unless these patterns were found in set. In addition, we also change the direction of original pattern to generate different string. For instance, if neither the string ***fdsa*** nor ***qwer*** appear in the training set but can be assigned a small probability after smoothing. At the same time, the probability of those patterns that have appeared will slightly decrease.

5. String Reorganization Module

In this section, we discover some special preferences for humans in constructing passwords by performing statistics on large-scale datasets. According to these conclusions, we build a string reorganization module based on the Markov chain [6], following on naturally from the previous module.

5.1 Distribution of String

We conduct experiments using the datasets in **Table 1** to get the character distribution and the basic password structure, shown in **Table 3**. **Table 3** shows character distribution. In all password datasets, letters and digits account for more than 97%. Therefore, the deeper habit of constructing passwords is hidden in the L and D segments of the basic password structure. There is an interesting phenomenon: In the English datasets, digits account for 27%, and letters account for 69%; In the Chinese datasets, letters account for 30% but digits account for

68%. The reason for this phenomenon may be that digits are easier to remember for Chinese users, and English is not the native language of Chinese users.

From the statistics in **Table 3**, we can see that users tend to use character of the same type to construct passwords, in which letters and digits account for the majority. Therefore, we utilize the string longer than 4 in L segments and D segments to train the Markov model. In this way, we reorganize the string in the training set, which not only takes full advantage of the user's habit of constructing string but also prevent overfitting.

Table 3. Distribution of basic segments

All	All-En	Rockyou	000webhost	Twitter		All-Ch	CSDN	Taobao	7k7k
D	27.71	29.55	24.34	27.56		69.15	69.21	68.41	69.76
L	71.77	69.87	75.28	71.95		30.49	30.33	31.02	29.89
S	0.52	0.58	0.38	0.49		0.46	0.46	0.57	0.35

5.2 Reorganizing String

After the structure partition module trains the password dataset, we can get different segments and their corresponding string. We find that letters and digits make up a large part of the password by separately counting the character types in the password. The specific statistical results will be described in detail in the next section. Therefore, we decide to use the string in L segment and D segment as training sets to train the Markov model. Since most of the string lengths are greater than 4, we employ 4-order Markov chain that performs best in trawling attacking to reorganize string.

In the 4-order Markov chain model, the probability of the next character appearing is based on the character string that is 3 in front of it. Therefore, for a given string, the probability of the 4-order Markov model are computed as follows:

$$P(c_1,\cdots,c_m) = P(c_1,\cdots,c_4)\cdot\prod_{i=4}^{m-1} P(c_{i+1} \mid c_{i-3},\cdots,c_i) \tag{5}$$

where the initial probability $P(c_1,\cdots,c_4)$ and the transition probability $P(c_{i+1} \mid c_{i-3},\cdots,c_i)$ are statistically obtained at the training stage. At the stage of string generating, the probability of each string in passwords is obtained by iterating the Equ. 5. In the end, the all new string for each segment is generated in descending order of probability.

6. BiLSTM Based Password Modification Module

In this section we try to explore the cross-site password behaviors for password reuse and modification. According to the statistics in [24-27], users often modify or reuse their existing passwords for easy memory when registering on a new website. These fragile password behaviors make it possible to guess user's unknown passwords using an existing password of this person for targeted attackers. Unfortunately, there is discrepancy in password reuse behaviors among different users, so how to quantify the user's behavior uniformly in grammar is a challenge. In order to solve this problem, we define 9 types of password modification rules. Further, our algorithm adopts a recurrent neural network classifier to predict the probability in using each modification, in the scene of knowing one of the user's website passwords.

6.1 Password Reuse & Modification

For a given pair of sister passwords, our goal is to determine if one of them is transformed by another and chase down the modification rules. **Fig. 4** shows the high-level workflow. Overall, we construct 9 password modification rules by manually examining 3,000 random password pairs and the results of previous studies [25-28]. These rules are tested on the CEM-dataset and EEM-dataset, and the results are presented in **Table 5**.

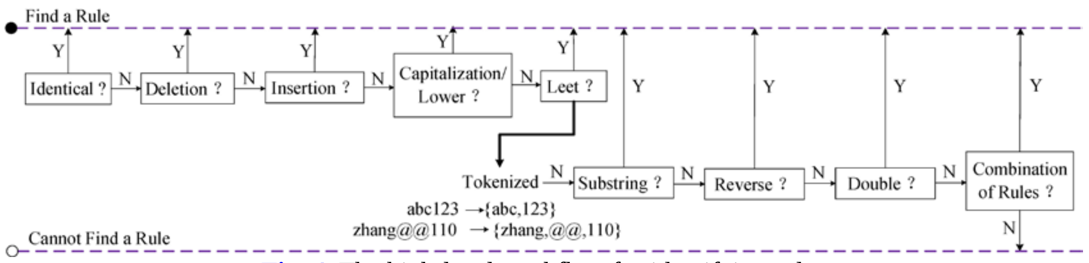


Fig. 4. The high-level workflow for identifying rules

Table 5. Distribution of modification rules

Rules	Ratio for EEM-dataset	Ratio for CEM-dataset
①. Identical	36.2%	44.6%
②. Deletion	8.5%	9.4%
③. Insertion	5.3%	6.1%
④. Capitalization/Lower	2.8%	2.3%
⑤. Leet	1.1%	1.4%
⑥. Substring	5.5%	7.6%
⑦. Reverse	< 0.1%	0.9%
⑧. Double	3.1%	1.8%
⑨. Combination of Rules	1.7%	1.6%
Can't Find any Rule	35.8%	24.3%

6.1.1 Password Reuse

Factually, due to the limited memory capacity of human brains, users often reuse their passwords between different websites. To intuitively analyze password reuse in practice, we perform a measurement on the CEM-dataset and EEM-dataset. For each user, we have two sister password to cross-check. Our counting only relates to whether the user has the behavior of password reuse. If there is a password reuse for the user, then the counting is increased by 1. In total, we find that 36.2% of English users and 44.6% of Chinese users have password reuse behavior.

6.1.2 Password Modification Rules

Studies [27,28] have investigated some popular modification rules for existing passwords. Different studies may give inconsistent conclusions about the specific use of these rules because of the size of the survey, the differences in the target population, and other reasons. But overall, the frequency to use these transformation rules is basically similar. In the actual user password settings, the similar password modification rules that may be used are too numerous to mention. Integrating various studies, we select the 9 types of transformation rules from them. Each rule is discussed in detail below:

1. Identical: For completeness, we consider reuse the same password as a special modification rule.

2. Deletion: Delete type of character whose number accounts for a small part of the password. The password commonly contains two or more character types. In addition, many users are accustomed to adding a character to the existing password when setting a new password. Therefore, this rule can also be deleting a character in the password. For example, ***password123*** can be transformed to get ***password***, ***password12*** or ***password1***. The passwords for this modification are usually longer or more complex. As shown in **Table 6**, we count the proportion of deleted position and deleted type in the password pairs with the deletion rule, then find that more than 90 percent of the modification occur at the tail of the password. We also find that more than 90% of the passwords delete the entire substring of the tail.

Table 6. The deletion pattern in two datasets (EEM-dataset, CEM-dataset)

Datasets	Delete Position	Ratio	Delete Type	Ratio
EEM-dataset	Tail	93.2%	Entire Substring	91.8%
	Head	5.6%	Last Character	3.5%
	Both Ends	1.2%	Last Two Characters	1.7%
CEM-dataset	Tail	95.4%	Entire Substring	94.8%
	Head	3.6%	Last Character	1.5%
	Both Ends	1.0%	Last Two Characters	0.7%

3. Insertion: In contrast to deletion, insertion rule adds characters to the original password. In order to make the modified password and the original password as similar as possible, users tend to insert a string at the tail and head in the original password. The inserted string is usually composed of the same type of character. As shown in **Table 7**, 94.7 percent of insertion occur at the tail of the password, most of the inserted characters are digits, and ***1***, ***123***, ***11***, and ***111*** add up to 44.4% of the total.

Table 7. The insertion pattern in two datasets (EEM-dataset, CEM-dataset)

Dataset	Insert Position	Ratio	Insert Length	Ratio
EEM-dataset	Tail	94.7%	1	61.6%
	Head	4.6%	2	18.4%
	Both Ends	0.7%	≥3	20.0%
	Insert Character	**Ratio**	**Insert String**	**Ratio**
	Digit	33.8%	a	27.2%
	Letter	62.2%	abc	11.6%
	Mixture	2.9%	1	4.8%
	Special Char	1.1%	12	3.5%
CEM-dataset	Tail	95.2%	1	64.3%
	Head	3.7%	2	16.6%
	Both Ends	1.1%	≥3	18.9%
	Digit	81.7%	1	32.2%
	Letter	12.1%	123	10.7%
	Mixture	3.9%	11	8.4%
	Special Char	2.7%	111	3.1%

4. Capitalization/Lower: This is one of the most common password modification rule that generally corresponds to the first letter transformation of the password. But it may also have the case that changes the entire letter segment (if a letter segment is all lower case, it will be converted to full capitalization. Otherwise, all of them are converted to lowercase).

5. Leet: This rule is mainly to replace certain characters with other similar-looking ones. We combine the previous studies to choose the 6 most popular leet rules: a$\leftrightarrow$ @, s$\leftrightarrow$ $, o$\leftrightarrow$ 0, i$\leftrightarrow$ 1, e$\leftrightarrow$ 3, t$\leftrightarrow$7. For instance, the ***password*** is changed to ***p@$$w0rd***. These 6 modification rules almost cover 95 percent of the leet pairs in EM-dataset.

6. Substring: If there is a substring of a special character in the password, the substring is used as a separator, swapping the strings on both sides of it. For passwords consisting of digits and letters, simply swap the digits and letters. For example, ***qwer1234*** can be transformed to get ***1234qw*** and ***zhang@@110*** can be transformed to get ***110@@zhang***.

7. Reverse: The reverse transformation usually occurs in the passwords that are shorter and contain fewer types of characters to facilitate memory. For example, ***abc123*** can be transformed to get ***321cba***.

8. Double: The short passwords are copied and then spliced behind itself to make the new password have the palindrome characteristics, which is easy to remember and looks safe. Users are accustomed to using this method. For instance, ***pass*** can be transformed to get ***passpass***.

9. Combination of Rules: In general, users may also choose multiple rules to modify their passwords. In all rules of **Table 5**, rules 2-5 modify the characters in the original password, but rules 6-8 change the password at the substring level without changing the original character. In our scenarios, we construct some combination rules based on these characteristics. **Table 8** shows the proportion of each rule used in all users with the combination of Rules in the Em dataset.

Table 8. The combination rules

Rule Combination	**Ratio**
Cap/Low + Insertion	32.6%
Cap/Low + Deletion	29.5%
Cap/Low + Double	13.1%
Leet + Substring	10.6
Cap/Low + Substring	5.4%
Cap/Low + Reverse	3.2%
Leet + Reverse	0.3%
Leet + Double	0.9%
Others	4.4%

In summary, some rules in the proposed 9 types can be directly used for password modification (e.g., 1, 4, 5, 6, 7, and 8 in **Table 5**), while others have no fixed modification method (e.g., 3, 4, 9 in **Table 5**). Therefore, we use the statistical results to limit the three types of modification rules, as shown in the **Table 9**. As for **Deletion**, we delete the entire substring at the tail. When it comes to **Insertion**, for English users, we insert the character ***a*** at the tail with the probability of 0.6, ***abc*** with the probability of 0.25, ***1*** with the probability of 0.1, and ***12*** with the probability of 0.05; For Chinese users, we insert ***1*** with the probability of 0.6, ***123*** with the probability of 0.2, ***11*** with the probability of 0.15, and ***111*** with the probability of 0.05. As for the mixing rule, we only consider the common four combinations types, Cap/Low + Insertion, Cap/Low + Deletion, Cap/Low + Double and Leet + Substring, with the probabilities of 0.38, 0.35, 0.15, 0.12, respectively.

Table 9. The restriction of deletion and insertion

Rule	Specific Operation	probability
Deletion	entire substring at the tail	1
Insertion (English)	a	0.6
	abc	0.25
	1	0.1
	12	0.05
Insertion (Chinese)	1	0.6
	123	0.2
	11	0.15
	111	0.05
Rule Comnination	Cap/Low + Substring	0.38
	Cap/Low + Reverse	0.35
	Cap/Low + Double	0.15
	Cap/Low + Substring	0.12

6.2 Establishing Model

In this subsection, we construct a rule classifier based on the BiLSTM recurrent neural network to automatically assign probability to each modification rule according to the characteristics of the password.

The BiLSTM-based rule classifier includes an input layer, several BiLSTM layers, a softmax layer, and an output layer. Among them, the BiLSTM is an improved model of LSTM [30], reading the input in both directions with 2 separate hidden layers: the forward and the backward. Two hidden states are $\vec{h}_t = LSTM(x_t, \vec{h}_{t-1})$ and $\overleftarrow{h}_t = LSTM(x_t, \overleftarrow{h}_{t-1})$. We summarize the information from the forward and the backward hidden states by concatenating them, i.e. $h_t = \left[\vec{h}_t, \overleftarrow{h}_t\right]$. We select the labeled Em-Dataset to train our model, as shown in **Fig. 5**. Firstly, transforming textual passwords into vector form so that the neural network can recognize by preprocessing, and then input them into the BiLSTM neural network layer to extract the feature vector of the passwords. Finally, the classification of the rules is achieved through the softmax multiple classifiers.

Before training, we translate the password into a vector pattern that the neural network can recognize. For example, given a n-length password $x_1 x_2 \cdots x_n$, for each character x_1 we use the $Con(\cdot)$ to represent this conversion: $Con(x_i) = \mathbf{a_i}$. Thus, the password $x_1 x_2 \cdots x_n$ is represented as $\mathbf{a_1 a_2 \ldots a_n}$.

At each time step $t (1 \le t \le n)$, BiLSTM layers extract features of $\mathbf{a_1 a_2 \ldots a_n}$, then the m-th Bi-LSTM layer outputs $h_m^{(t)}$. The softmax layer calculates the probability belonging to each category based on the parameter h_θ passed from the previous layer. Compared with the category label, calculate the correct rate and then adjust the parameters θ and δ by using Back Propagation Through Time (BPTT). For each training password, the cost function of softmax regression is as follows:

$$J(\theta, \delta_1, \delta_2, \cdots, \delta_m) = -\frac{1}{m}\left[\sum_{i=1}^{n}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T h_m^i}}{\sum_{l=1}^{k} e^{\theta_j^T x^{(i)}}}\right] \quad (6)$$

Where θ is the parameter of softmax layer and δ is the parameter of BiLSTM layers. k

represents the number of password modification rules, which is 9 in this article.

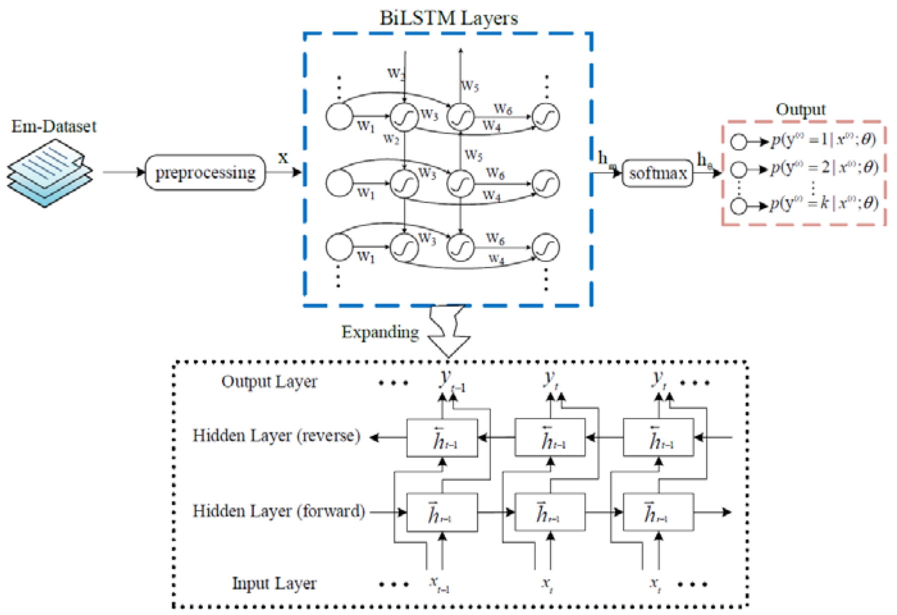


Fig. 5. The BiLSTM classification model.

7. Experiment

The experimental software and hardware operating environment in this article is as follows: Intel Core i7-4790 CPU, 16GB RAM, GTX 1070 GPU, Python2.9, Matlab2015a, 64-bit Ubuntu 16.04 LTS OS.

7.1 Experiment Setup

In the experiment, we compare the targeted password attacking algorithm based on structure partition and string reorganization (denoted as TG-SPSR) with the mainstream four algorithms: Markov, PCFG, Personal-PCFG, TarGuess-I. Among them, Markov and PCFG belong to trawling attacking algorithm, in the training process, only the password is required. The other two are targeted attacking algorithms that need to provide the user's personal information. However, there are only three original datasets containing complete PII (12306, Rootkit, Clixsense), and the number of users is limited. Thus, we build two new PII-associated datasets, one of them by matching Taobao with 51job and 12306 using E-mail address, another similarly by matching Twitter with Clixsense. Finally, we obtained five datasets containing complete PII information, as shown in the **Table 10**.

Table 10. Basic info about five datasets with complete PII

Dataset	Language	Account
PII-Taobao	Chinese	315,620
12306	Chinese	128,793
PII-Twitter	English	132,150
Rootkit	English	69,418
Clixsense	English	2,154,338

Using the datasets in **Table 10**, we set up 9 attacking scenarios. The first 4 attacking scenarios are for Chinese datasets, the others are for English, as shown in **Table 11**. In order to verify the attack effect when the training set and the test set come from the same website, we set the attacking scenario 1, 2, 3. 80 percent of the original dataset as training set and the remaining as testing set. In the rest of the attacking scenarios, both the training sets and the testing sets come from different websites.

Table 11. Basic info about cracking scenarios

scenarios	**Training set → testing set**
# 1	PII-Taobao → PII-Taobao
# 2	PII-Twitter → PII-Twitter
# 3	Clixsense → Clixsense
# 4	PII-Taobao → 12306
# 5	PII-Twitter → Clixsense
# 6	Clixsense → Rootkit
# 7	12306 → PII-Taobao
# 8	Clixsense→PII-Twitter
# 9	Rootkit→Clixsense

Specifically, we choose the RNN with 2 hidden BiLSTM layers and 256 neurons per BiLSTM in the following experiments to train the classifier. In addition, the Markov order in the string reorganization module and the compared algorithm is set to 4.

7.2 Experiment Result

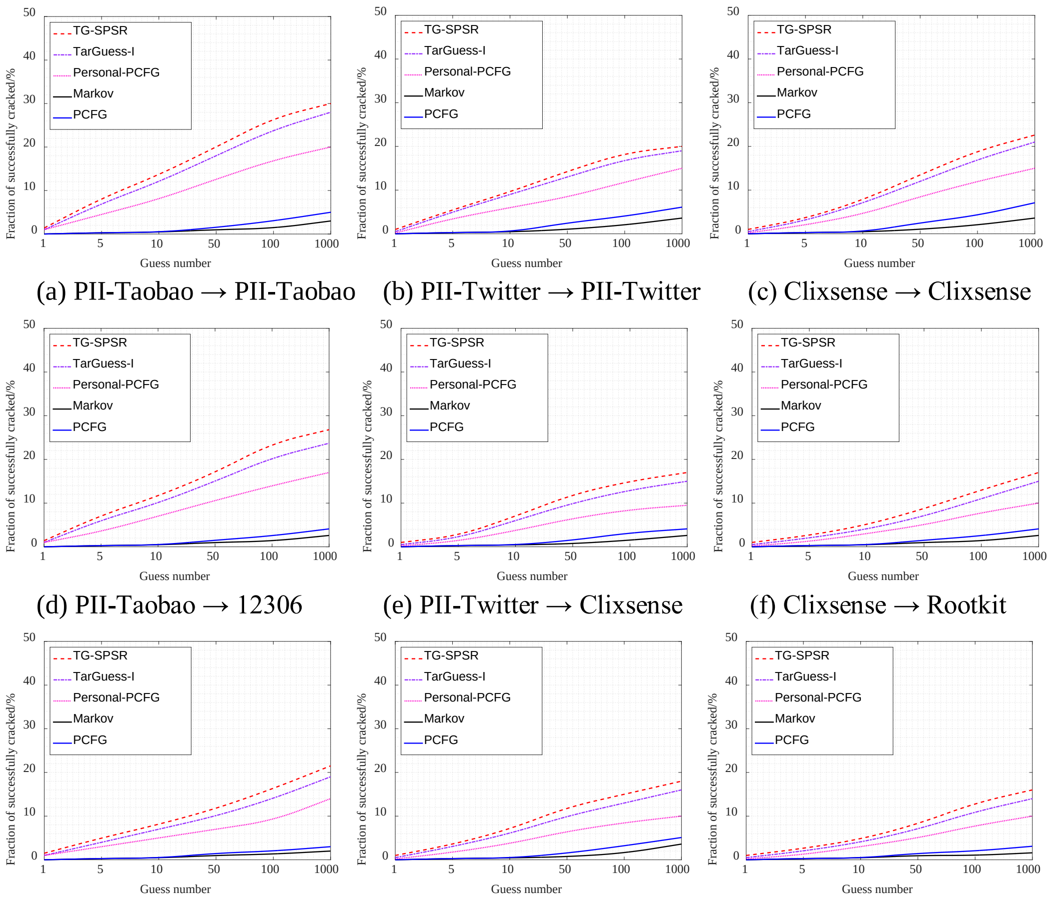
In all 10 experiments conducted in this section, the training process can be completed in 5 minutes on an ordinary personal computer, and it takes less than 30 minutes to generate one hundred million guesses for each user. Therefore, we use the guessing-number-graph to evaluate the effectiveness of our algorithm and compare it with the existing four mainstream password attacking algorithms (PCFG, Markov, personal-PCFG, TarGuess-I). In **Fig. 6**, we set the number of guessing within 1000, the number in **Figs. 7** is between 10^3 and 10^8.

Further, in order to demonstrate the advantages of the TG-SPSR model in targeted password attacking, we also test the average guess number to crack each password in different attacking scenarios in **Table 11**. We stipulate that the attacker can guess up to 100 times when cracking the password. If the password is not cracked within the specified guess number, it is not counted in the statistics. Finally, calculate the average guess number according to Equ.7.

$$AveGuessNum = \frac{\sum_{i=1}^{N} Guessnum(i)}{N} \tag{7}$$

Where N is the total number of passwords cracked in the test set and *Guessnum(i)* is the guess number for the ith password.

In the case where the same private information PII available to attacker and within 1000 guesses, **Figs. 6 (a)~(i)** show that: (1) TG-SPSR outperforms two trawling attacking algorithms, averaging about 196% on the Chinese datasets and 154% on the English datasets. (2) Compared with two targeted attacking algorithm, TG-SPSR outperforms Personal-PCFG by 60% and 10% for TarGuess-I when training set and testing set are from the same website. 84% for Personal-PCFG and 15% for TarGuess-I when training set and testing set are from different websites.



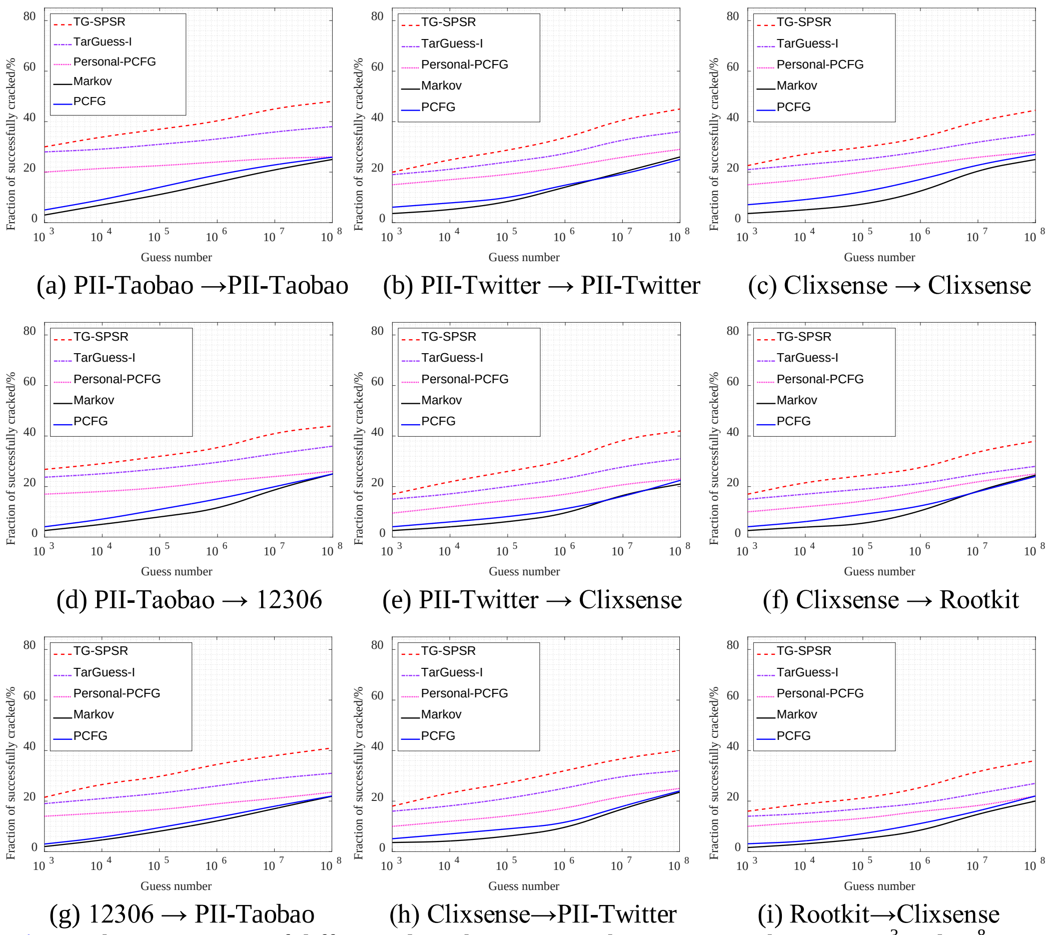
(a) PII-Taobao ⟶ PII-Taobao (b) PII-Twitter ⟶ PII-Twitter (c) Clixsense ⟶ Clixsense

(d) PII-Taobao ⟶ 12306 (e) PII-Twitter ⟶ Clixsense (f) Clixsense ⟶ Rootkit

(g) 12306 ⟶ PII-Taobao (h) Clixsense ⟶ PII-Twitter (i) Rootkit ⟶ Clixsense

Fig. 6. The success rate of different algorithms in 9 attacking scenarios within 1000 guesses.

Under the same private information PII available to attacker and within one hundred million guesses, **Figs. 7 (a)~(i)** show that: (1) Compared with two trawling attacking algorithm, **TG-SPSR** outperforms them by 120% on the Chinese datasets and by 95% on the English datasets when the number of guesses is less than 1000, in contrast, outperforms them by 98% on the Chinese datasets and by 65% on the English datasets. (2) Compared with two targeted attacking algorithm, if training set and testing set are from the same website, **TG-SPSR** outperforms Personal-PCFG by 71% and 19% for TarGuess-I when the number of guesses less than 10^6, and 80% for Personal-PCFG and 26% for TarGuess-I when the number of guesses more than 10^6. (3) If training set and testing set are from different websites, TG-SPSR outperforms Personal-PCFG by 79% and 28% for TarGuess-I when the number of guesses less than 10^6, and 90% for Personal-PCFG and 36% for TarGuess-I when the number of guesses more than 10^6.



(a) PII-Taobao →PII-Taobao (b) PII-Twitter → PII-Twitter (c) Clixsense → Clixsense

(d) PII-Taobao → 12306 (e) PII-Twitter → Clixsense (f) Clixsense → Rootkit

(g) 12306 → PII-Taobao (h) Clixsense→PII-Twitter (i) Rootkit→Clixsense

Fig. 7. The success rate of different algorithms in 9 cracking scenarios between 10^3 and 10^8 guesses.

In the case where the same private information PII available to attacker and limit the maximum guess number to 100, **Fig. 8** shows that: (1) Compared with two trawling attacking algorithm (Markov, PCFG), the average guess number for the three targeted attacking algorithms (TG-SPSR, TarGuess-I, Personal-PCFG) is much smaller ; (2) Compared with the other two targeted attacking algorithm (TarGuess-I, Personal-PCFG), TG-SPSR performs better. In the nine cracking scenarios, the average guess number for the TG-SPSR algorithm is less than 30, which is 15%-20% less than TarGuess-I and 35%-45% less than Personal-PCFG.

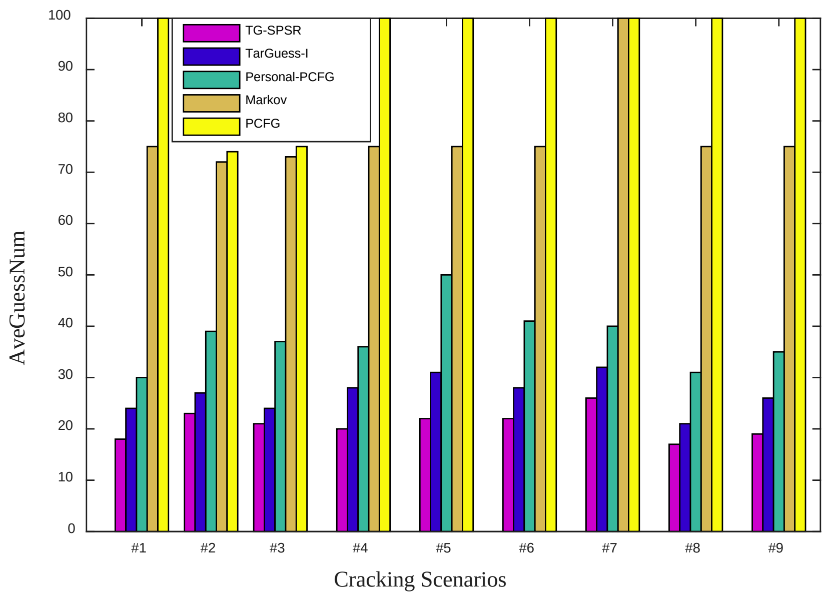


Fig. 8. Average guess number in different cracking scenarios

In summary, we can draw the following conclusions: (1) Chinese users prefer to employ personal information in their passwords; (2) In the targeted online attacking scenarios (limited number of guesses), TG-SPSR algorithm performs extremely better than the traditional trawling attacking algorithms and also has advantages over its foremost counterparts (Personal-PCFG, Targuess-I), which indicates that TG-SPSR can better use the known private information to mine user password construction habits. (3) In the targeted offline attacking scenario (unlimited number of guesses), with the increase of guess number, the advantage of TG-SPSR compared to the traditional trawling attacking algorithms will be reduced, but still outperforms them by 50%, and the advantage will be magnified compared to its foremost counterparts, which fully shows that our algorithm is more generalizable.

8. Conclusion

For the first time, we integrated the PCFG and Markov chain algorithms in the targeted password attacking, and proposed a complete attacking model (TG-SPSR), which contains three modules: structure partition module, string reorganization module, rule-based password generation module. Extensive experimental results show that our model performs extremely better than traditional trawling attacking algorithms and is more generalizable compared with its foremost counterparts.

In the further work, we will continue to optimize our algorithm, at the same time, we will also tap deeper password structure features and private information, and explore better methods of structure partitioning. For example, in the structure partitioning module, only five kinds of PII are considered. However, the experiments of Wang et al.. showed that non-identifiable private information such as gender and hobbies also affect the user's behavior. How to quantify this effect in the password structure is a question worth pondering.

References

[1] Ping Wang, Ding Wang, Xinyi Huang, "Advances in password security," *Computer Research and Development*, vol. 53, no. 10, pp. 2173-2188, 2016. Article(CrossRef Link).

[2] J. Bonneau, C. Herley, P. V. Oorschot, et al., "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, pp. 78-87, 2015. Article(CrossRef Link).

[3] C. Herley, P. V. Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Security & Privacy*, vol. 10, no. 1, pp. 28-36, 2012.

[4] D. Freeman, M. Dürmuth, B. Biggio, "Who are you? a statistical approach to measuring user authenticity," in *Proc. of the Network & Distributed System Security Symposium*, pp. 1-15, February 21-24, 2016. Article(CrossRef Link).

[5] J. Yan, A. Blackwell, R. Anderson and Grant A, "password memorability and security: empirical results," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 25-31, 2004. Article(CrossRef Link).

[6] A. Narayanan, V. Shmatikov, "Fast dictionary attacks on passwords using time-space trade off," in *Proc. of 12th ACM conference on Computer and communications security*, pp. 364-372, October 16-18, 2005. Article(CrossRef Link).

[7] M. Weir, S. Aggarwal, B. D. Medeiros, et al., "Password cracking using probabilistic context-free grammars," in *Proc. of 30th IEEE Symposium on Security and Privacy*, pp. 391-405, May 17-20, 2009. Article(CrossRef Link).

[8] J. Ma, W. N. Yang, M. Luo, et al., "A study of probabilistic password models," in *Proc. of 35th IEEE Symposium on Security and Privacy*, pp. 689-704, May 18-21, 2014. Article(CrossRef Link).

[9] M. Dürmuth, F. Angelstorf, C. Castelluccia, et al., "OMEN: Faster password guessing using an ordered markov enumerator," in *Proc. of 7th International Symposium on Engineering Secure Software and Systems*, pp. 119-132, March 4-6, 2015. Article(CrossRef Link).

[10] S. Houshmand, S. Aggarwal, R. Flood, "Next gen PCFG password cracking," *IEEE Transactions on Information Forensics & Security*, vol. 10, no. 8, pp. 1776-1791, 2015. Article(CrossRef Link).

[11] D. Wang, Z. J. Zhang, P. Wang, et al., "Targeted online password guessing: an underestimated threat," in *Proc. of 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1242-1254, October 24-28, 2016. Article(CrossRef Link).

[12] All Data Breach Sources, May, 2016. Article(CrossRef Link).

[13] Turkey: personal data of 50 million citizens leaked online, April 2016.

[14] Amid Widespread Data Breaches in China, Dec, 2011. Article(CrossRef Link)

[15] Y. Zhang, F. Monrose, and M. Reiter, "The security of modernpassword expiration:an algorithmic framework and empirical analysis," in *Proc. of ACM CCS*, pp. 176-186, October 4-8, 2010. Article(CrossRef Link).

[16] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *Proc. of NDSS*, pp. 23-26, February 23-26, 2014. Article(CrossRef Link).

[17] Y. Li, H. Wang, and K. Sun, "A study of personal informationin human-chosen passwords and its security implications," in *Proc. of IEEE Inform*, pp. 1-9, April 10-14, 2016. Article(CrossRef Link).

[18] Nearly 80 percent of Internet users suffer identity leaks, July, 2015. Article(CrossRef Link).

[19] Four Years Later, Anthem breached again: hackers stole credentials, Feb. 2015. Article(CrossRef Link).

[20] A. Grimes. Roger, "Password size does matter[EB/OL]," July 2006. Article(CrossRef Link).

[21] R. Shay, S. Komanduri, A. Durity, et al., "Designing password policies for strength and usability," ACM Transactions on Information and System Security, vol. 18, no. 4, pp. 1-34, 2016. Article(CrossRef Link).

[22] J. Bonneau, C. Herley, P. C. Van Oorschot, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, pp. 78-87, 2015. Article(CrossRef Link).

[23] C. Castelluccia, A. Chaabane, M. Dürmuth, et al., "When privacy meets security: leveraging personal information for password cracking," *Computer science*, 2013. Article(CrossRef Link).

[24] A. Singer, W. Anderson, R. Farrow, "Rethinking password policies," *Usenix and Sage*, vol. 38, pp. 14-18, 2013. Article(CrossRef Link).

[25] S.M. Haque, M. Wright, and S. Scielzo, "A study of user password strategy for multiple accounts," in *Proc. of 3th ACM Conference on Data and Application Security and Privacy*, pp. 173-176, 2013. Article(CrossRef Link).

[26] R. Wash, E. Rader, R. Berman, and Z. Wellmer, "Understanding password choices: how frequently entered passwords are reused across websites," in *Proc. of Symposium on Usable Privacy and Security*, pp. 175-188, June 22-24, 2016. Article(CrossRef Link).

[27] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: an algorithmic framework and empirical analysis," in *Proc. of 17th ACM Conference on Computer and Communications Security*, pp. 176-186, 2010. Article(CrossRef Link).

[28] S. Pearman, J. Thomas, P. E. Naeini, et al., "Let's go in for a closer look: observing passwords in their natural habitat," in *Proc. of ACM Sigsac Conference on Computer and Communications Security*, pp. 295-310, 2017. Article(CrossRef Link).

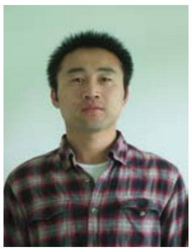
[29] S. Hochreiter, J. Schmidhuber, "Long short-tem memory," *Neural computation*, Vol. 9, no. 8, pp. 1735-1780, 1997. Article(CrossRef Link).

Mengli Zhang, born in 1993, M. S. candidate. His research interests include password security and big data security.

Qihui Zhang, born in 1983, Ph. D. , lecturer. Her main research focuses on big data security.

Liu Wenfen, born in 1965. PhD, professor, PhD supervisor. Her research interests include cryptography and information security

Hu Xuexian, born in 1982. PhD, associate professor. His current research interests include security protocol and big data security.



Wei Jianghong, born in 1987. PhD, lecturer. His main research interests include authentication protocol, big data security and privacy protection.