This is the version submitted to ACM CCS 2016, mainly with some typos corrected, and we are preparing the Camera Ready version according to the reviewers' constructive suggestions. July 25th, 2016.

Targeted Online Password Guessing: An Underestimated Threat

Ding Wang[†], Zijian Zhang[†], Ping Wang[†], Jeff Yan[*], Xinyi Huang[‡]
[†]School of EECS, Peking University, Beijing 100871, China
[*]School of Computing and Communications, Lancaster University, United Kingdom
[‡]School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China
{wangdingg, zhangzj, pwang}@pku.edu.cn; jeff.yan@lancaster.ac.uk; xyhuang81@gmail.com

ABSTRACT

While *trawling* online/offline password guessing has been intensively studied, only a few studies have examined *targeted online guessing*, where an attacker guesses a specific victim's password for a service, by exploiting the victim's personal information such as one sister password leaked from the victim's another account and some personally identifiable information (PII). A key challenge for targeted online guessing is to choose the most effective password candidates, while the number of guess attempts allowed by a server's lockout or throttling mechanisms is typically very small.

We propose *TarGuess*, a framework that systematically characterizes typical targeted guessing scenarios with seven sound mathematical models, each of which is based on varied kinds of data available to an attacker. These models allow us to design novel and efficient guessing algorithms. Extensive experiments on 10 large real-world password datasets show the effectiveness of TarGuess. Particularly, TarGuess I∼IV capture the four most representative scenarios and *within* 100 guesses: (1) TarGuess-I outperforms its foremost counterpart by 142% against security-savvy users and by 46% against normal users; (2) TarGuess-II outperforms its foremost counterpart by 169% on security-savvy users and by 72% against normal users; and (3) Both TarGuess-III and IV gain success rates over 73% against normal users and over 32% against security-savvy users. TarGuess-III and IV, for the first time, address the issue of cross-site online guessing when given the victim's one sister password and some PII.

Keywords

Password authentication; Targeted online guessing; Personal information; Password reuse; Bayesian theory.

1. INTRODUCTION

Passwords firmly remain the most prevalent mechanism for user authentication in various computer systems. To understand their security, a number of probabilistic guessing models, such as probabilistic context-free grammars (PCFG) [27] and Markov n-grams [18], have been suggested. A common feature of these guessing models is that they characterize a *trawling offline guessing* attacker who mainly works against the leaked password files and aims to crack as many accounts as possible. Recent research [6, 14] has realized that, it should be the role of websites to protect user passwords from offline guessing by securely storing user password files, while common users only need to choose passwords that can survive online guessing.

Online guessing attacks can be launched against the publicly facing server by *anyone* using a browser at *anytime*, with the primary constraint being the number of guesses allowed. *Trawling online guessing* mainly exploits users' behavior of choosing popular passwords [19, 26], and it can be well addressed by various security mechanisms at the server (e.g., suspicious login detection, rate-limiting and lockout [13]). However, *targeted online guessing* (see Fig. 1) can exploit weak popular passwords, passwords reused across sites, and passwords containing personal information. This is a serious security concern, since various Personally Identifiable Information (PII) and leaked passwords become readily available [1, 2, 15]. For instance, the most recent large-scale PII data breach in April 2016 [2] involves 50 million Turkish citizens, accounting for 64% of the population. According to the CNNIC 2015 report [28], over 78.2% of the 668 million Chinese netizens have suffered PII data leakage; In a series of recent breaches, over 253 million American netizens become victims of PII and password leakage [22].

The main challenge for targeted online guessing is to effectively characterize an attacker $\mathcal{A}$'s guessing model, with useful information available well captured, while the guess number allowed is small – the NIST Authentication Guideline [16] requires Level 1 and 2 systems to keep login failures less than 100 per user account in any 30-day period. The following explains why it is a challenge.

First, people's password choices vary much among each other. When creating a password, some people reuse an existing password, some modify an existing password. Some people incorporate PII into their passwords, others do not; Some favor digits, some favor letters, and so on. Thus, a user population's passwords created for a given service can differ greatly. Therefore, the trawling guessing models [18, 21, 24, 27], which aim to produce a *single* guess list for *all* users, are not suitable for characterizing targeted guessing.

Second, users' PII is highly heterogeneous. Some kinds of PII (e.g. name, and hobby) are composed of letters, some (e.g. birthday and phone num) are composed of digits, and some (e.g. user name) are a mixture of letters, digits and symbols; Some PII (e.g. name, birthday and hobby), as shown in Fig. 2, can be directly used as password components, while others (e.g. gender and education) cannot. As we will show, most of them have an impact on people's

DOI: 10.1145/1235

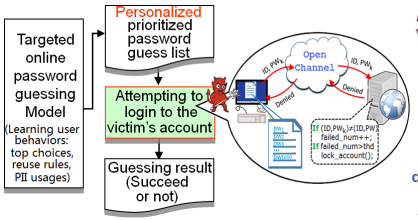


Fig. 1: Targeted online guessing.

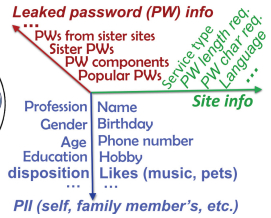


Fig. 2: Multiple info for $\mathcal{A}$.

password choice. Thus, it is challenging to, at a large-scale, automatically incorporate such heterogeneous PII into guessing models when the guess attempts allowed is limited.

Third, users employ a diversified set of transformation rules to modify passwords for cross-site reuse. As shown in [11, 25], when given a password, there are over a dozen transformation rules, such as insert, delete, capitalization and leet (e.g., `password`→ `passw0rd`) and the synthesized ones (e.g., `password`→ `Passw0rd1`), that a user can utilize to create a new password. How to prioritize these rules for each individual user is not easy.

Moreover, which transformation rules users will apply for password reuse are often context dependent. Suppose attacker $\mathcal{A}$ targets Alice's eBay account which requires passwords of length 8^+, and knows that Alice is in her 30s. With access to a sister password `Alice1978Yahoo` leaked from Alice's Yahoo account, $\mathcal{A}$ will have a higher chance by guessing `Alice1978eBay` than by `Alice1978`. Yet, when Alice's leaked password is `123456`, $\mathcal{A}$ would more likely succeed by guessing `Alice1978` than by `Alice1978eBay`. When site password policies are also considered, the situation may further vary. Such context dependence necessitate an adaptive, semantics-aware cross-site guessing model.

1.1 Related work

Zhang et al. [29] suggested a algorithm for predicting a user's future password with *previous ones* for *the same account*. Das et al. [11] studied the password reuse issue, and proposed a cross-site password cracking algorithm However, their algorithm is *not optimal* for targeted online guessing for four reasons. First, it did not consider common popular passwords (e.g., `iloveyou`, `pa$$w0rd`) which do not involve with password reuse or user PII. Second, it assumes that all users employ the transformation rules in a fixed priority. Yet, as we observe, this priority is actually dynamic and context-dependent. Third, their algorithm does not consider various synthesized rules. Fourth, it is heuristics based.

Li et al. [17] examined how user's PII may impact password security, and found that 60.1% of users incorporate at least one kind of PII into their passwords. They proposed a semantics-rich algorithm, Personal-PCFG, which considers six types of personal info: name, birthdate, phone number, National ID, email address and user name. However, as we will show, its *length-based* PII matching and substitution approach makes it inaccurate to capture user PII usages, greatly hindering the cracking efficiency.

1.2 Our contributions

In this work, we make the following key contributions:

- **A practical framework**. To overcome the challenges discussed above, we propose *TarGuess*, a framework to characterize typical targeted online guessing attacks, with sound probabilistic models (rather than ad hoc models or heuristics). TarGuess captures seven typical scenarios, with each based on a different combination of various information available to the attacker.
- **Four probabilistic algorithms**. To model the most representative targeted guessing scenarios, we propose four algorithms by leveraging probabilistic techniques including PCFG, Markov and Bayesian theory. Our algorithms all significantly outperform prior art. We further show how they can be readily employed to deal with the other three remaining attacking scenarios.
- **An extensive evaluation.** We perform a series of experiments to demonstrate that both the efficacy and general applicability of our algorithms. Our empirical results show that an overwhelming fraction of users' passwords is vulnerable to our targeted online guessing. This suggests that the danger of this threat has been significantly underestimated.
- **New insights.** For example, Type-based PII-tags are more effective than length-based PII-tags in targeted guessing; Simply incorporating many kinds of PII into algorithms will *not* increase success rates, which is counter intuitive; The success rate of a guess decreases with *a power law* as the guess number increases.

2. PERSONAL INFORMATION

The most prominent feature that differentiates a targeted guessing attack from a trawling one is that, the former involves user-specific data, or so-called "personal info". This term is sometimes used inter-changeably with the term "personally identifiable info" [9, 17], while sometimes their definitions vary greatly in different situations, laws, regulations [20, 23]. Generally, a user's personal info is "any info relating to" this user [23], and it is broader than PII. For better comprehension, in Table 1 we provide the first classification of personal info in the case of password cracking, making a systematical investigation of targeted guessing possible.

We divide user personal info into three kinds, with each kind having varied degree of secrecy, different roles in passwords and various types of specific elements. The first kind is user PII (e.g., name and gender), which is natively semipublic: public to friends, colleagues, acquaintances, etc., yet private to strangers; The second kind is user identification credentials, and parts of them (e.g., user name) are public, while parts of them (e.g., password) are exclusively private; The remaining user personal data falls into the third kind and is irrelevant to this work. We further divide user PII into two types: Type-1 and Type-2. Type-1 PII (e.g., name and birthday) can be the building blocks of passwords, while Type-2 PII (e.g., gender and education [19]) may impact user behavior of password creation yet cannot be directly used in passwords. Each type of PII shapes our guessing algorithms quite distinctly.

Here we highlight a special kind of user personal information — a user's passwords at various web services. As shown in [11, 25], users tend to reuse or modify their existing passwords at other sites (called *sister passwords*) for new accounts. However, a user's sister password(s) is becoming more and more easily available due to the unending catactrophic leakages of passwords lists from high-profile sites (see [1, 3, 15]).

As there is a messy mixture of multiple dimensions of info (see Fig. 2) potentially available to the attacker $\mathcal{A}$, it is challenging to characterize $\mathcal{A}$. We tackle this issue by assuming that all the public info (e.g., leaked PW lists and site policies) should be available to $\mathcal{A}$, and then by defining a series of attacking scenarios (see Table 2) based on varied types of U's personal info given to $\mathcal{A}$. This is reasonable: (1)

Table 1: Explication of user personal info (NID stands for National identification number, e.g., SSN; PW for password)

Different kinds of personal info		Degree of secrecy	Roles in PWs	**Considered** in this work(✓)	Not Considered in this work(×)
Personally identifiable information (PII)	Type-1	Semipublic	Explicit	Name, Birthday, Phone num, NID	Place of birth, Likes, Hobbies, etc.
	Type-2	Semipublic	**Implicit**	Gender, Age, Language	Faith, Disposition, Education, etc.
User identification credentials		Private	Explicit	Passwords, PINs	Finger prints, Private keys, etc.
		Public	Explicit	User name, Email address	Debit card number, Health IDs, etc.
Other kinds of personal data		—	—	—	Employment, Financial records, etc.

Table 2: A summary of the four most representative scenarios of targeted online guessing

Attacking scenario	Exploiting *public* info (e.g., datasets and policies)	Exploiting user *personal* info †			Existing literature	**Our model**
		One sister PW	Type-1 PII	Type-2 PII		
Trawling #1	✓				Ref. [18, 21, 27]	
Targeted #1	✓		✓		Ref. [17]	TarGuess-I
Targeted #2		✓			Ref. [11]	
	✓	✓			None	TarGuess-II*
Targeted #3	✓	✓	✓		None	TarGuess-III
Targeted #4	✓	✓	✓	✓	None	TarGuess-IV

*As public password datasets are readily available, TarGuess-II and [11] is comparable because they exploit the same type of user PII.
†A total of $7(=C_3^1+C_3^2+C_3^3)$ scenarios result from combining the 3 types of personal info. With TarGuess-I∼IV, all 7 cases are tackled in Sec.4.

$\mathcal{A}$ is smart and likely to exploit the readily available public info to increase her chance; and (2) $\mathcal{A}$ would use different attacking strategies when given different personal info.

Note that, here we only consider scenarios where $\mathcal{A}$ is with at most one sister password of user U. The reason is that, among the 547.56M of leaked password accounts that we have collected over a period of six years, less than 1.02% (resp. 1.73%) of them have more than one match by email (resp. user name). Similarly, among the 7.96M accounts collected by Das et al. [11], only 152 (0.00191%) of them have more than one match by email. Therefore, *it is realistic to assume that most users have leaked one sister password, and $\mathcal{A}$ can exploit U's this sister password for attacking.*

3. HUMAN BEHAVIORS OF PASSWORD CREATION

We first report a large-scale empirical study of human behaviors in creating passwords, in particular, how often they choose popular passwords, how often to reuse passwords, how often to make use of their own PII.

Table 3: Basic info about our 10 password datasets

Dataset	Web service	Language	When leaked	Total PWs	With PII
Dodonew	E-commerce	Chinese	Dec., 2011	16,258,891	
CSDN	Programmer	Chinese	Dec., 2011	6,428,277	
126	Email	Chinese	Dec., 2011	6,392,568	
12306	Train ticketing	Chinese	Dec., 2014	129,303	✓
Rockyou	Social forum	English	Dec., 2009	32,581,870	
000webhost	Web hosting	English	Oct., 2015	15,251,073	
Yahoo	Web portal	English	July, 2012	442,834	
Rootkit	Hacker forum	English	Feb., 2011	69,418	✓
Xiaomi*	Mobile, cloud	Chinese	May, 2014	8,281,385	
Xato	Synthesised	English	Feb., 2015	9,997,772	

*Xiaomi passwords are in salted-hash and will be used as real targets.

3.1 Our datasets

Our evaluation builds on ten large real-world password datasets (see Table 3), including five from English sites and five from Chinese sites. They were hacked by attackers or leaked by insiders, and disclosed publicly on the Internet, and some of them have been used in trawling password models [12, 18, 25]. Rootkit initially contains 71,228 passwords hashed in MD5, and we recover 97.46% of them by using our TarGuess-IV and various trawling guessing models [18, 24] in one week. In total, these datasets consist of 95.83 million plain-text passwords and cover various popular web services. The role of each dataset will be specified in Sec. 5.

Table 4: Basic info about our personal-info datasets

Dataset	Language	Items num	Types of PII useful for this work
Hotel	Chinese	20,051,426	Name, Gender, Birthday, Phone, NID
51job	Chinese	2,327,571	Email, Name, Gender, Birthday, Phone
12306	Chinese	129,303	Email, User name, Name, Gender, Birthday, Phone, NID
Rootkit	English	69,324	Email, User name, Name, Age, Birthday

Particularly, two of these 10 password datasets are with various types of PII as shown in Table 4. Besides, we further employ two auxiliary PII datasets, aiming to augment the password datasets by matching the email address to facilitate a more comprehensive understanding of the role of PII in user-chosen passwords. To the best of knowledge, *our corpus is the largest and most diversified ever collected for evaluating the security threat of targeted online guessing.*

3.2 Popular passwords

Table 5 shows how often users from different services choose popular passwords. It is disturbing that 0.79%∼10.44% of user passwords can be guessed by just using the top 10 passwords. Generally, Chinese passwords are more concentrated than English ones, which may imply that the former would be more prone to online guessing. Most of the top Chinese passwords are only made of simple digits, while popular English ones tend to be meaningful letter strings or keyboard patterns. Love plays an important role — `iloveyou` and `princess` are among the top-10 lists of two English sites, while `5201314`, which sounds as "I love you forever and ever" in Chinese, is among the top-10 lists of three Chinese sites. Other factors such as culture (see `18881888`) and site name (see `rockyou` and `rootkit`) also show their impacts on password creation.

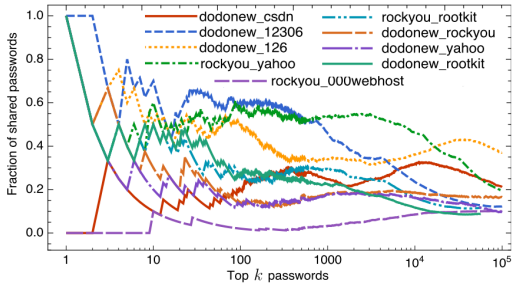


Fig. 3: Fraction of PWs shared between two sites.

Fig. 3 illustrates the fraction of top-k passwords shared between two different services with varied thresholds of k. Generally, the fraction of shared passwords from the same language is substantially higher than that of shared passwords from different languages; The fraction of shared passwords between any two services is less than 60% at any threshold k larger than 10. This accords with [25] and shows that, both language and service play an important role in shaping users' top popular passwords.

Rockyou and 000webhost share significantly fewer common passwords than other pairs do. We examine into these two datasets and find that 99.29% of 000webhost passwords include *both* letters and digits, indicating that this site enforces such a password creation policy. This can also be corroborated by Table 5 where all top-10 000webhost passwords are composed of both letters and digits. Similarly, we find that CSDN requires passwords to be of length 8^+.

Table 5: Top-10 most popular passwords of each service

Rank	Dodonew	CSDN	126	12306	Rockyou	000webhost	Xato	Yahoo	Rootkit
1	123456	123456789	123456	123456	123456	abc123	123456	123456	123456
2	a123456	12345678	123456789	a123456	12345	123456a	password	password	password
3	123456789	11111111	111111	**5201314**	123456789	12qw23we	12345678	welcome	**rootkit**
4	111111	**dearbook**	password	123456a	password	123abc	qwerty	ninja	111111
5	**5201314**	00000000	000000	111111	**iloveyou**	a123456	123456789	abc123	12345678
6	123123	123123123	123123	**woaini1314**	**princess**	123qwe	12345	123456789	qwerty
7	a321654	1234567890	12345678	123123	1234567	secret666	1234	12345678	123456789
8	12345	**88888888**	**5201314**	000000	**rockyou**	YfDbUfNjH10305070†	111111	sunshine	123123
9	000000	111111111	**18881888**	qq123456	12345678	asd123	1234567	**princess**	qwertyui
10	123456a	147258369	1234567	1qaz2wsx	abc123	qwerty123	dragon	qwerty	12345
% of top-10	3.28%	10.44%	3.52%	1.28%	2.05%	0.79%	1.46%	1.01%	3.94%

†The letter-part (i.e., `YfDbUfNjH`) can be mapped to a Russian word which means "navigator". Why it is so popular is beyond our comprehension.

3.3 Password reuse

While users have to maintain probably several times as many password accounts as they did 10 years ago, human-memory capacity remains stable. As a result, users tend to cope by reusing passwords across different services [14, 25]. Several empirical studies [4, 11] have explored the password reuse behaviors of English and European users, yet as far as we know, no empirical results have been reported about Chinese users, who reach 668 million by Dec., 2015 [10] and account for over 25% of the world's Internet population.

To fill this gap, we intersect 12306 with Dodonew by matching email, and further eliminate the users with identical password pairs. This produces a new list `12306&Dodonew` with two non-identical sister passwords for each user. Similarly, we obtain two more intersected Chinese password lists and three intersected English lists (see Fig. 4). During the matching process, we find that 34.02%∼71.11% of Chinese users' sister PW pairs are identical (and eliminated), while these figures for English users are 6.25%∼21.96% (see Sec. 5.1). This suggests that our English users reuse less.

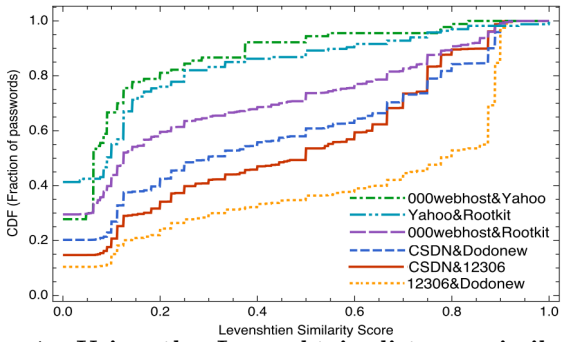


Fig. 4: Using the Levenshtein-distance similarity metric to measure the similarity of two passwords chosen by the same user across different services. Most users modify passwords in a non-trivial way.

We employ the widely accepted Levenshtein-distance metric to measure the similarity between two different passwords of a given user. Fig. 4 shows that, sister passwords of Chinese users generally have higher similarity than English users under both metrics, implying that Chinese users modify PWs less complexly. About 30% of the non-identical Chinese PW pairs have similarity scores in the range of [0.7, 1.0], while this figure for our English pairs is less than 20%. This implies that *the majority of users modify passwords in a non-trivial approach, and it would be challenging to model users's such transformation behaviors.*

A plausible reason for the observation that our English users reuse less and modify passwords more complexly is that, Rootkit is a hacker forum and 000webhost is mainly used by website administrators, and thus *the users of both sites are likely to be more security-savvy than normal users.* Thus, the lists `Rootkit&000webhost`, `Rootkit&Yahoo` and `000webhost&Yahoo` would show more secure reuse behaviors than that of normal English and Chinese users.

In 2014, Das et al. [11] found that the fraction of identical sister password pairs of common English users is 43%, which roughly accords with our Chinese users yet 2∼6 times higher than our English users. They also showed that about 30% of their non-identical English password pairs have similarity scores in [0.7, 1.0], well according with our Chinese users. Moreover, the survey results on password reuse behaviors of common Chinese users [25] are largely consistent with the survey results on common English users [11]. Both empirical and survey results suggest that *common Chinese and English users have similar reuse behaviors, while our English users would be good representatives of security-savvy users.*

3.4 Password containing personal info

We show in Table 6 how often users employ their *own* PII to build passwords. Since some password lists have no PII (see Table 3), we correlate them with the PII datasets in Table 4 by matching email, producing seven PII-associated password lists which are much more diversified than [17]. As expected, highly heterogeneous PII becomes components of passwords, and users like to use names, birthdays and their variations. Particularly, a non-negligible fraction of users employ just their full names (0.75%∼1.87%) as passwords, and 1.00%∼5.16% of Chinese users use just their birthdays as passwords. Surprisingly, email and user name prevail in passwords of both groups of users, ranging from 0.77% to 5.07% and from 0.54% to 2.34%, respectively. In comparison, English users exhibit a more secure behavior in PII usages, for our English users represent security-savvy ones.

Fig. 7 shows the impact of type-2 PII : (1) passwords of Dodonew female users are more concentrated; (2) passwords of Dodonew users in age≤24 and age≥46 have similar length distributions (pairwise χ^2 test, p-value= 0.009), while users in age 25∼45 are significantly different (p-values$<10^{-6}$). Similar results are in all other datasets.

Summary. Our PII-associated password corpus is so far the largest and most diversified ever collected for evaluating targeted online guessing. Particularly, it, for the first time, covers (security-savvy) English users. While users' three vulnerable behaviors might be potentially exploited to improve cracking, our results show that varied circumstances (e.g., language, service and policy), non-trivial transformation rules and highly heterogeneous PII all would make it a challenging task to *automate* this process, especially when given a limited guessing number (e.g., 100 by NIST [7, 16]).

4. TARGUESS: A FRAMEWORK FOR TARGETED ONLINE GUESSING

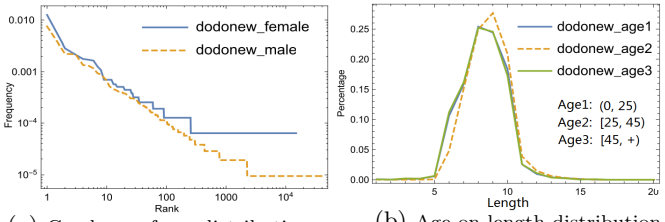
We now propose *TarGuess*, a practical framework that effectively addresses the realistic yet challenging problem of modeling various targeted online guessing scenarios.

As shown in Fig. 5, TarGuess consists of three phases (i.e., preparing, training and guessing). The design of the first and third phases is straightforward, and the main task lies in the second one. TarGuess captures four types of most

Table 6: Percentages of users building passwords with (and only with) their *own* heterogeneous personal info[†]

Typical usages of personal info (examples)	PII-Dodonew (161,510)		PII-126 (30,741)		PII-CSDN (77,439)		PII-12306 (129,303)		PII-Rootkit (69,330)		PII-Yahoo (214)		PII-000webhost(2,950)	
Full_name (`lei wang`, `john smith`)	4.68	0.82	3.00	1.32	4.85	1.81	5.02	1.13	1.38	0.75	2.34	1.87	2.44	1.32
Family_name (`wang`, `smith`)	11.15	0.01	6.16	0.00	9.75	0.00	11.23	0.00	2.28	0.78	4.67	1.87	3.73	1.46
Given_name (`lei`, `john`)	6.49	0.07	4.10	0.12	6.26	0.08	6.61	0.07	0.49	0.07	0.93	0.00	0.75	0.20
Abbr. full_name (`wl`, `lwang`, `js`, `jsmith`)	13.64	0.02	6.36	0.00	9.42	0.00	13.13	0.00	0.15	0.01	0.00	0.00	0.20	0.00
Birthday(19820607, 06071982, 07061982)	3.12	1.00	3.70	2.77	6.29	5.16	4.33	1.77	0.08	0.06	0.47	0.00	0.10	0.07
Year of bithday (1982)	8.92	0.00	8.84	0.01	11.37	0.00	10.78	0.00	0.75	0.01	1.40	0.00	1.12	0.00
Date of bithday (0607, 0706)	8.32	0.00	10.48	0.02	11.84	0.00	10.03	0.00	0.44	0.01	0.47	0.00	0.58	0.00
Abbr. bithday(198267, 671982, 761982, 820607, 060782)	2.37	0.59	2.60	1.71	2.89	1.45	3.31	1.12	0.10	0.05	0.00	0.00	0.20	0.14
Family_name+bithday (`wang19820607`, `smith06071982`)	0.08	0.08	0.05	0.05	0.03	0.03	0.14	0.14	0.00	0.00	0.00	0.00	0.00	0.00
Family_name+Abbr. bithday①(`wang198267`, `smith671982`)	0.11	0.11	0.03	0.02	0.05	0.05	0.15	0.14	0.00	0.00	0.00	0.00	0.00	0.00
Family_name+Abbr. bithday②(`wang820607`, `smith060782`)	0.17	0.17	0.07	0.07	0.13	0.11	0.17	0.16	0.00	0.00	0.00	0.00	0.00	0.00
Family_name+year of birth (`wang1982`, `smith1982`)	0.55	0.22	0.20	0.07	0.22	0.07	0.64	0.25	0.01	0.00	0.00	0.00	0.00	0.00
Family_name+date of birth (`wang0607`, `smith0607`)	0.12	0.09	0.05	0.03	0.08	0.04	0.16	0.12	0.01	0.00	0.00	0.00	0.00	0.00
User name (`icemoon12`, `bluebirdz`)	1.54	1.14	0.54	0.38	0.61	0.43	1.96	1.32	1.59	0.92	2.34	1.40	2.20	1.32
Email_prefix (l0veu4ever@example.com)	5.07	3.07	2.52	1.60	4.35	2.48	3.03	1.82	0.77	0.44	4.21	1.87	1.32	0.78
Phone_num (11-digit Chinese mobile num 13511336677)	0.10	0.10	0.48	0.45	0.50	0.45	0.07	0.01	—	—	—	—	—	—
'a'+birthday(`a19820607`, `a06071982`, `a07061982`)	0.16	0.13	0.04	0.02	0.03	0.02	0.16	0.12	0.00	0.00	0.00	0.00	0.00	0.00
Full_name+1 (`wanglei1`, `johnsmith1`)	1.49	0.22	0.51	0.03	0.84	0.03	1.65	0.17	0.06	0.01	0.00	0.00	0.03	0.00

[†]All the decimals in the table use '%' as the unit. For instance, 4.68 in the top left corner means 4.68% of the 161,510 Dodonew users (associated with PII) employ their full name to build passwords; 0.82 means 0.82% of the 161,510 Dodonew users' passwords are *just* their full names.



(a) Gender on freq. distribution. (b) Age on length distribution.

Fig. 7: Impact of type-2 PII on password creation. Both gender and age show tangible impact.

representative targeted online guessing scenarios, with each type based on varied kinds of personal info available to $\mathcal{A}$ (see Table 2): (i) only type-1 PII; (ii) one sister password; (iii) combination of *i* and *ii*; (iv) combination of *iii* and type-2 PII. To model these four scenarios, we suggest four guessing models (I∼IV) by leveraging a number of probabilistic techniques such as PCFG, Markov and Bayesian theory. We also show that, with GarGuess-I∼IV, the three remaining scenarios can also be well addressed.

4.1 TarGuess-I

TarGuess-I aims to online guess a user U's passwords by exploiting U' some type-1 PII (e.g., name and birthday, not gender). It builds on Weir et al.'s PCFG-based algorithm [27] which has been shown a great success in dealing with *trawling* guessing scenarios. In the training phase of [27], each password is seen as a combination of letter(L)-, digit(D)- and symbol(S)- segments. For example, `loveyou@1314` is parsed into the L-segment "loveyou", S-segment "@" and D-segment "1314", and its base structure is $\mathsf{L}_7\mathsf{S}_1\mathsf{D}_4$; `wanglei@1982` is also parsed into $\mathsf{L}_7\mathsf{S}_1\mathsf{D}_4$.

Our new algorithm. To capture PII semantics, besides the L, D, S tags as with PCFG [27], we introduce a number of *type-based* PII tags (e.g., N_1∼N_7 and B_1∼B_{10}). For a *type-based* PII tag, its subscript number stands for *a particular sub-type* of one kind of PII usages but not the *length* matched, as opposed to the L, D, S tags. For instance, N stands for name usages, while N_1 for the usage of full name, N_2 for the abbr. of full name (e.g., `lw` from "`lei wang`"), ⋯; B stands for birthday usages, B_1 for full birthday in YMD format (e.g., 19820607), B_2 for full birthday in MDY, ⋯. This gives rise to a PII-enriched context-free grammar $\mathcal{G}_I = (\mathcal{V}, \Sigma, \mathcal{S}, \mathcal{R})$, where:

1) $\mathcal{S} \in \mathcal{V}$ is the start symbol.
2) $\mathcal{V} = \{\mathcal{S}; \mathsf{L}, \mathsf{D}, \mathsf{S}; \mathsf{N}_1, \cdots, \mathsf{N}_7; \mathsf{B}_1, \cdots, \mathsf{B}_{10}; \mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3; \mathsf{E}_1, \mathsf{E}_2, \mathsf{E}_3; \mathsf{P}_1, \mathsf{P}_2; \mathsf{I}_1, \mathsf{I}_2, \mathsf{I}_3;\}$ is a finite set of variables,[1] where: (1) N_1 and N_2 have been specified earlier, N_3 for family name (e.g., `wang`), N_4 for given name, N_5 for the 1st letter of the given name + family name (e.g., `lwang`), N_6 for last name+ the 1st letter of the given name (e.g., `wangl`), N_7 for family name with its 1st letter capitalized (e.g., `Wang`); (2) B_1 and B_2 have been specified earlier, B_3 stands for full birthday in DMY (e.g., `07061982`), B_4 for the date in birthday, B_5 for the year in birthday, B_6 for Year+Month (e.g., (e.g., `198206`), B_7 for Month+Year (e.g., `061982`), B_8 for the last two digits of year + date in MD format (e.g., `820607`), B_9 for date in MD format + the last two digits of year (e.g., `060782`), B_{10} for date in DM format + the last two digits of year (e.g., `070682`); (3) A stands for account name usages, A_1 for full account name (e.g., `icemoon12`), A_2 for the (first) letter-segment of account name (e.g., `icemoon`), A_3 for the (first) digital-segment of account name (e.g., `12`); (4) E stands for email prefix usages, E_1 for the full email prefix (e.g., `loveu1314` from `loveu1314@example.com`), E_2 for the first letter-segment of email prefix (e.g., `loveu`), E_3 for the first digital-segment of account name (e.g., `1314`); (5) P stands for mobile phone num usages, P_1 for the full num, P_2 for the first three digits, P_3 for last four digits; (6) I stands for Chinese government ID, I_1 for the last 4 digits, I_2 for the first 3 digits, I_3 for the first 6 digits.[2]
3) Σ={95 printable ASCII codes, *Null*} is a finite set disjoint from $\mathcal{V}$ and contains all the terminals of $\mathcal{G}_I$.
4) $\mathcal{R}$ is a finite set of rules of the form $A \to \alpha$, with $A \in \mathcal{V}$ and $\alpha \in \mathcal{V} \cup \Sigma$ (see Fig. 6).

A probabilistic context-free grammar (PCFG) is a CFG that, for a specific left-hand side (LHS) variable (e.g., L_4), all the probabilities associated with its rules (e.g., $\mathsf{L}_4 \to$ `love` and $\mathsf{L}_4 \to$ `Suny`) can add up to 1 [27]. It is not difficult to see that, the training phase of TarGuess-I can indeed automatically derive a PCFG. Using this grammar, in the guess generation process (see Fig. 6) we can further derive: (1) all the terminals (e.g., `love@1314`) which are instantiated from base structures that only consist of L, D and S tags; and (2) all the pre-terminals (e.g., $\mathsf{N}_3\mathsf{B}_5$ and $\mathsf{N}_3$1234) which are intermediate guesses consisting of PII-based tags. The final guess candidates come from these terminals as well as from instantiating all the pre-terminals with the victim's PII.

[1]The number of PII-based variables and their specific definitions depend on the nature of the PII to be trained (e.g., phone num in US is 10 digits while 11 digits in China) and on the granularity the attacker $\mathcal{A}$ prefers (e.g., $\mathcal{A}$ may prefer 4 types of name usages but not 7 as we do). Here we give a typical definition for attacking Chinese users, and it is easily tailored to other user groups.

[2]Our definitions are gained by recursively adjusting and training on the PII-associated 126 dataset, and to keep the basic machine learning principle, hereafter 126 will never be used as the test set.

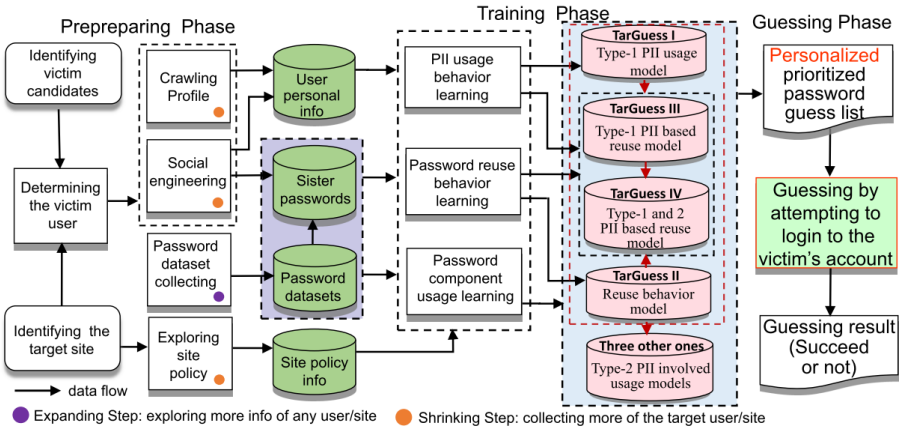


Fig. 5: An architectural overview of the TarGuess.

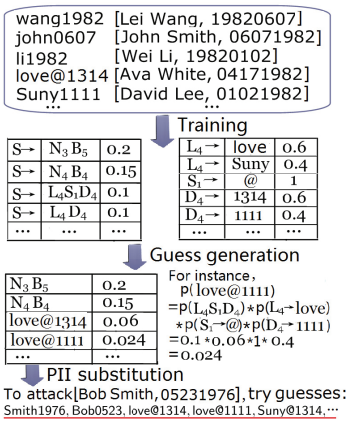


Fig. 6: TarGuess-I: an illustration.

Note that, to improve accuracy, we match in the longest-prefix rule and also only consider PII-segments with $len \geq 2$. For example, if `john06071982` matches John Smith's account name "john0607", it will be parsed into A_1B_5 using the longest-prefix rule, but not N_3B_2. In addition, we have only considered full MMDD dates in the definition of $B_1 \sim B_{10}$, yet many users tend to use an abbr. of date when possible (e.g., "198267" instead of "19820607"). Thus, when matching a birthday-based segment in the training phase, if an abbr. happens, the tag related to the corresponding full segment will be counted by one; In the password generation process, both full and abbreviated date segments will be produced. For instance, both "john06071982" and "john671982" will be produced if the structure N_3B_2 is used for guess generation.

A salient feature of $\mathcal{G}_I$ is that it is highly adaptive. On the one hand, whenever we want to consider new semantic usages (e.g., website name) or new type-1 PII usages (e.g., hobby), we can simply define new corresponding *type-based* tags (e.g., W for website name and H for hobby) as the same when we define the N and B tags. In the training and guess generation phases, all the operations related to H and W are similar with that of N and B tags. It is "Plug-and-Play". On the other hand, even if TarGuess-I defines the B tag yet the training set has no birthday info, $\mathcal{G}_I$ still works properly—it will not parse passwords using B tags and simply parse birthday info in passwords using the D tag. That is, $\mathcal{G}_I$ is "self-dumping" and we do not need to specially eliminate the B-related tags in such cases.

An independent study. In a recent paper (published in April 2016), Li et al. [17] presented a *length-based* PII matching method. Our work is independent from theirs.

They introduced six kinds of PII tags: N for name, B for birthdate, E for email, A for account name (user name), P for phone number, and I for NID. In contrast to our type-based approach, each PII-based tag in [17] uses a subscript number to denote the length *len* of the matched PII segment (only $len \geq 3$ are considered in [17]), following the same notations of the L, D and S tags as in Weir et al [27]. As a result, in [17], `wanglei@1982` now is parsed into the N segment "wanglei", S segment "@" and B segment "1984", and its base structure will be $N_7S_1B_4$; "`loveyou@1314`" is parsed into $L_7S_1D_4$. Within 100 guesses, their "Personal-PCFG" algorithm cracked about 17% of their test dataset by using "perfect dictionaries".

However, we discovered a weakness in their Personal-PCFG. Their algorithm uses *length-based* tags, the same as Weir et al.'s algorithm [27]: it differentiates a segment's length, but is insensitive in a segment's subtype. For example, both `john@1982` and `wang@1982` will be parsed into $N_4S_1B_4$, because both "`wang`" and "`john`" are of length 4, despite the fact that "`wang`" is a family name while "`john`" is a given name. As shown in Fig. 8, this not only introduces both under-estimations and over-estimations in the training phase, but also leads to illogical situations in the guess generation phase. In the training phase, since "`wang`", "`smith`" and "`li`" are all family names and "`1982`" a user's year of birth, the probability of N_4B_4 shall be 0.6 but not 0.4, the probability of L_2B_4 shall be 0 but not 0.2; In the guess generation phase, "`lee`" is of length 3, but there is no base structure N_3B_4 available, and thus the guess `lee1982` will be given a probability 0 and thus not be generated.

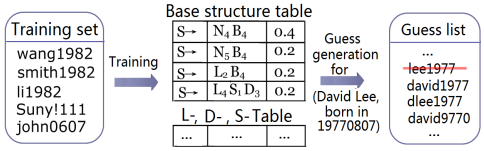


Fig. 8: A weakness of Personal-PCFG [17].

According to our grammar $\mathcal{G}_I$, both `wang1982` and `li1982` will be parsed into the same base structure N_3B_5, `john1982` is parsed into N_4B_5, and thus the guess `lee1982` can be generated using the base structure N_3B_5 for "David Lee" born in 1982. This addresses the weakness in [17].

Evaluation. For fair comparison, we leverage the 12306 dataset as with [17] and follow their experimental setups (see Fig. 9) as closely as possible. The only exception is that, we do not use "the perfect (L-) dictionary" which is collected directly from the *test* set, because this not only introduces overfitting issues [18, 27] but also is unrealistic in practice. Instead, all our experiments directly learn the L- dictionary from the 12306 *training* set, a recommended practice in password cracking [12,18]. Fig. 9 shows that, within $10\sim10^3$ guesses, our TarGuess-I *outperforms* Persona-PCFG [17] by 37.11%∼73.33%, and *outperforms* the three trawling online guessing algorithms [5, 18, 27] by at least 412%∼740%.

We have further examined to what extent each individual PII would impact TarGuess-I. As shown in Fig. 9, within 100 guesses, our TarGuess-I can successfully crack a 12306 user's password with an average chance of 20.26% when given this user's email, account name, name, birthday, phone num and NID; this figure is 20.18% when given email, account name, name and birthday; this figure is 13.61% when given name and birthday; this figure is 6.04% when given only name. Our results suggest that *email, account name, name and birthday would be very valuable for an online attacker*, while phone num and NID provide marginally improved success rates. Interestingly, Personal-PCFG [17] exploits

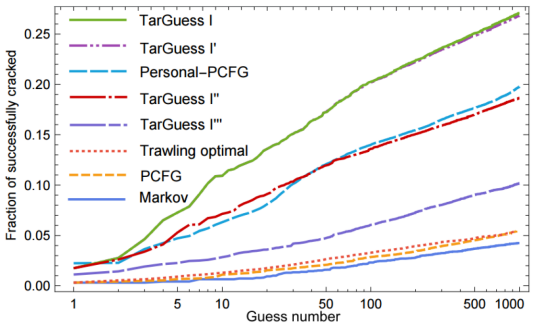


Fig. 9: A comparison of TarGuess-I with Personal-PCFG [17] and other versions of TarGuess-I, trained on the 50% of 12306 dataset and tested on the remaining 50%. Both TarGuess-I and Personal-PCFG [17] employ six kinds of the 12306 type-1 PII, while TarGuess-I′ eliminates phone num and NID, TarGuess-I″ further eliminates email and user name, and TarGuess-I‴ further eliminates birthday.

two more PII attributes than TarGuess-I′ yet is much less effective, suggesting that simply incorporating more info into algorithms will not always yield more effectiveness.

Summary. We are the first to use type-based PII tags for building a semantics-aware grammar. Within 10^2 guesses, our TarGuess-I has a success of 20.18% when given a 12306 user's email, user name, name and birthday. Within $10{\sim}10^3$ guesses, TarGuess-I outperforms Personal-PCFG [17] by cracking 37.11%∼73.33% more passwords.

4.2 TarGuess-II

TarGuess-I aims to online guess a user U's password PW_x at one service (e.g., CSDN) when given U's one sister password PW_s leaked from another service (e.g., Dodonew). This is a challenging task for two reasons. Firstly, the online guessing number allowed is small. Secondly, there are over a dozen transformation rules, such as insert, delete, capitalize and leet (e.g., `password`→ `passw0rd`) and the synthesized ones (e.g., `password`→ `Passw0rd`), at users' choices to create PW_x by modifying PW_s. This process depends on the password creation policy, the value of service and each user's creativity. Moreover, even if $\mathcal{A}$ knows that U is likely to insert three digits, which digital sequence will be *exactly* used by U? It is worth noting that, users also love to use top popular passwords instead of modifying PW_s (see Sec. 3.2). The sole work by Das et al. [11] considers cross-site guessing by using U's one sister password, yet it is ineffective due to four reasons as we have shown in Sec. 1.

In this work, we prefer a data-driven approach. We use two lists of passwords as training sets, one leaked from a similar policy/service with the target site, the other is similar with that of PW_s, and look for the same users in these two lists by matching email. Further, the identical PW pairs are eliminated, and this creates a new list of non-identical sister PW pairs $\{PW_A, PW_B\}$. Then, we measure how PW_B is modified from PW_A, or whether PW_B is simply a popular password. To determine whether PW_B is a popular one, we build a top-10^4 list $\mathcal{L}$ for the target service (e.g., CSDN) from various leaked lists with consideration of policy and language, e.g., $\mathcal{L}$=$\{pw$ |len(pw)≥ 8 and the value of $P_{csdn}(pw) * P_{126}(pw) * P_{dodonew}(pw)$ ranks top-$10^4\}$.

As shown in Fig. 10, in the training phase of TarGuess-I, first of all it determines whether $PW_B \in \mathcal{L}$ or not. If yes, the occurrence of $PW_B \in \mathcal{L}$ increases by one; If not, the pair (PW_A, PW_B) first goes through the structure-level training and then the segment-level training. In the structure-level process, TarGuess-II first parses passwords with L, D, S tags as with TarGuess-I. For instance, `abc123` is parsed into $\mathsf{L}_3\mathsf{D}_3$. According to [11, 25], we consider six main types of structure-level transformation rules: insertion (e.g., $\mathsf{L}_3\mathsf{D}_3$→ $\mathsf{L}_3\mathsf{D}_3\mathsf{S}_2$), deletion (e.g., $\mathsf{L}_3\mathsf{D}_3\mathsf{S}_2$)→$\mathsf{L}_3\mathsf{D}_3$), capitalization C, leet L, substring movement SM (e.g., `abc123`→`123abc`) and reverse R (e.g., `abc123`→`321cba`). There are two segment-level rules: insertion (e.g., $\underline{\mathsf{L}_3 : \text{abc}}$→ $\underline{\mathsf{L}_4 : \text{abcd}}$) and deletion (e.g., $\underline{\mathsf{L}_3 : \text{abc}}$ → $\underline{\mathsf{L}_2 : \text{bc}}$). For each of these eight types of rules, there exist a number of sub-types and we consider the most common ones. More specifically, for both levels of insertion (see Fig. 11), there are tail insertion ti (resp. ti') and head insertion hi (resp. hi'); For both levels of deletion, there are tail deletion td(resp. td') and head deletion hd(resp. hd');[3] For capitalization, there are four types as in Table 7; For leet, we consider 5 sub-types as in Table 8; For reverse, we consider 2 sub-types as in Table 10.

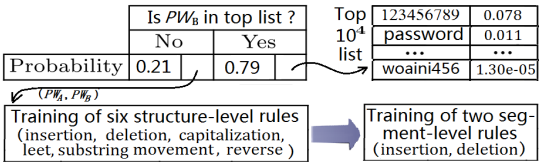


Fig. 10: The training process of TarGuess-II

The first step of the structure-level training is to employ the Levenshtein-distance (LD) algorithm (with only insertion and deletion enabled) to measure the similarity score d_1=LD(PW_A, PW_B) between PW_A and PW_B. Then, we use each structure-level rule (except for insertion and deletion) in the C, L, R, SM order to obtain PW'_A based on PW_A, when considering that their popularity order is C>L>SM>R [11, 25] and that the rule R would result in a more drastic change than SM. Upon each rule, we compute d_2=LD(PW'_A, PW_B). If d_2>d_1, such a rule is called a "live" one, then PW_A is updated to PW'_A, and the occurrence of the corresponding rule (see Tables 7 to 10) increases by one.

Upon all these live rules, assume PW''_A will be created from the original PW_A. To avoid futile transformations to dilute the effective ones, we require that if LD(PW''_A, PW_B) is smaller than a predefined threshold (e.g., 0.5 as suggested in this work), then all these "live" rules are *un*-counted, and the training process switches to the *next* password pair. Otherwise, both PW''_A and PW_B are parsed with L, D and S tags to be, e.g., $\mathsf{L}_4\mathsf{D}_3\mathsf{S}_2$ and $\mathsf{L}_6\mathsf{S}_1$. Since we do not consider the length of a PW segment in the structure-level, $\mathsf{L}_4\mathsf{D}_3\mathsf{S}_2$ and $\mathsf{L}_6\mathsf{S}_1$ will be seen as LDS and LS. Now we use the LD metric to compute d_3=LD("LDS″, "LS″) and meanwhile, the LD algorithm returns a LD edit route which records how to arrive "LS" from "LDS": first use the rule td on the S_2-segment, then use the rule td on the D_3-segment, and finally use the rule ta on the S_1-segment, producing PW'''_A with a base structure $\mathsf{L}_4\mathsf{S}_1$. Accordingly, the occurrence of all the corresponding items in Fig. 11(a) is updated.

Now we come to the segment-level training phase, and the focus is in the inner of the L-, D- or S- segment of a password. For PW'''_A (whose structure is $\mathsf{L}_4\mathsf{S}_1$) and PW_B $\mathsf{L}_6\mathsf{S}_1$, we use the LD metric to measure the similarity of their L-segments. As with the structure-level training, the LD metric is used to update the occurrence of all the corresponding rules in Fig.

[3]Combining our insertion and deletion operations can transform `abc123` to `abc!123`, achieving the middle insertion.

Table 7: Training of capitalization C (C_1: Cap. the 1st letter; C_2: Cap. all; C_3: Lower 1st; C_4: Lower all)

	No	C_1	C_2	C_3	C_4
Probability	0.95	0.03	0.01	0.007	0.003

Table 8: Training of the leet transformation rule L

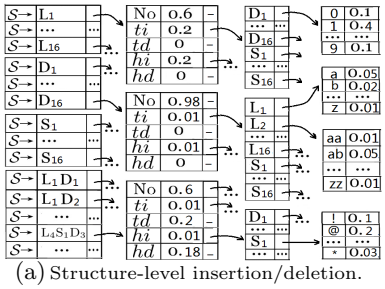
	No	$L_1: a \leftrightarrow @$	$L_2: s \leftrightarrow \$$	$L_3: o \leftrightarrow 0$	$L_4: i \leftrightarrow 1$	$L_5: e \leftrightarrow 3$
Prob.	0.95	0.02	0.01	0.01	0.005	0.005

Table 9: Training of substring movement

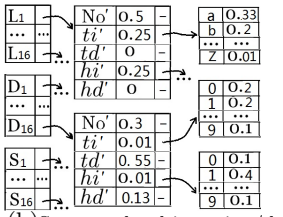
	substring moved SM	
	Yes	No
Prob.	0.03	0.97

Table 10: Training of reverse R (R_1: Reverse all; R_2: Reverse each segment)

	No	R_1	R_2
Probability	0.97	0.02	0.01



(a) Structure-level insertion/deletion.



(b) Segment-level insertion/deletion. The right two rows is better trained using n-grams.

Fig. 11: Training of two levels of insertion and deletion. As over 99% of PWs are with $len \leq$16 [18], only segments with $len \leq$16 are considered. The right-most two rows in (a) is better trained by PCFG [27] on a 10^6-sized PW list.

11(b). In our experiments, we find that the probabilities in the right-most row of Fig. 11(b) are better by computing using Markov n-grams [18] which are trained on a million-sized large password list, than by using the training as stated above. This is mainly because the size of the non-identical PW pairs in our training sets is only moderate and may lead to the sparsity issue. Fortunately, Markov n-gram model on million-sized PW lists can overcome this issue.

Our above two training phases give rise to a password-reuse based context-free grammar $\mathcal{G}_{II} = (\mathcal{V}, \Sigma, \mathcal{S}, \mathcal{R})$, where:

1) $\mathcal{V} = \{\mathcal{S}; \mathsf{L}, \mathsf{D}, \mathsf{S}; L, R, C, SM; ti, td, hi, hd; ti', td', hi', hd'\}$ is a finite set of variables.
2) $\mathcal{S} \in \mathcal{V}$ is the start symbol.
3) Σ={95 printable ASCII codes; C_1,$\cdots$,C_4; R_1,R_2; L_1, $\cdots$, L_5; `Yes`, `No`} is a finite set disjoint from $\mathcal{V}$.
4) $\mathcal{R}$ is a finite set of rules of the form $A \rightarrow \alpha$, with $A \in \mathcal{V}$ and $\alpha \in \mathcal{V} \cup \Sigma$ (see Fig. 11 and Tables 7 to 10).

Note that, $\mathcal{G}_{II}$ is a probabilistic context-free grammar due to the fact that, for a specific left-hand side (LHS) variable (e.g., $R \rightarrow$), of $\mathcal{G}_{II}$, all the probabilities associated with its rules (e.g., $R \rightarrow$`No`, $R \rightarrow R_1$ and $R \rightarrow R_2$) can add up to 1. Using $\mathcal{G}_{II}$, in the guess generation phase we can create a list of guesses with possibilities. For instance, when given `password`, Pr("`Pa$$word123`")= Pr($\mathcal{S} \rightarrow \mathsf{L}_8$)* Pr($\mathsf{L}_8 \rightarrow$ ta)* Pr($ta \rightarrow \mathsf{D}_3$)* Pr($\mathsf{D}_3 \rightarrow$123) *$P(C \rightarrow C_1)$ *Pr($L \rightarrow L_2$) *Pr($L \rightarrow L_2$) *Pr($R \rightarrow$`No`)*Pr($SM \rightarrow$`No`) =1 * 0.1 * 0.15 * 0.08 * 0.03 * 0.01 * 0.01 * 0.97 * 0.97 = 3.39 * 10^{-9}, where the related probabilities are referred to Tables 7 to 10. Then, all the probabilities of guesses generated by $\mathcal{G}_{II}$ should be multiplied by the factor α. This α represents the fraction of users who do *not* choose top passwords (e.g., 0.21 in Fig. 10). Then, the probability of each password in the top-10^4 list are multiplied by $1-\alpha$. Finally, these two lists are merged and sorted in decreasing order, and then we select the top k (e.g., k=10^3) as the final guess candidates.

In Fig. 12, we provide a comparison of TarGuess-II with Das et al.'s algorithm [11]. When given a user U's Dodonew password, within 100 guesses, Das et al. [11] gained a success rate of 8.98% against U's CSDN account, while the figure for TarGuess-II is 20.19%, reaching a 124.83% improvement. In a series of 10 experiments in Sec. 5, under the same personal info and within 100 guesses, TarGuess-II outperforms their algorithm [11] by 8.12%∼300% (avg. 111.06%). One may conjecture that the two variations of TarGuess-II employ more personal info than one sister PW, and thus they are more powerful. In what follows, we, for the first time, provide more than anecdotal evidence to back this conjecture.

4.3 TarGuess-III

TarGuess-III aims to online guess a user U's passwords by exploiting U's one sister password *as well as* some PII. This is realistic: if $\mathcal{A}$ wants to target U and knows U's one sister password, it is likely that $\mathcal{A}$ can also obtain some PII (e.g., email, name) about U. As far as we know, no public literature has ever paid attention to this kind of attacking scenario. Here we mainly consider type-1 PII (e.g., name), while type-2 PII (e.g., gender) will be dealt with in Sec. 4.4.

Given a limited guess number, more info available to TarGuess-III generally means more messy things to be considered and thus more challenges to be addressed. Suppose $\mathcal{A}$ wants to target Alice Smith's account at eBay which requires passwords to be of length 8^+, and knows Alice was born in 1978 and one of her passwords `Alice1978Yahoo` leaked from Yahoo. For the three guesses `Alice1978eBay`, `Alice1978` and `12345678`, which one shall $\mathcal{A}$ try first? If Alice's leaked password is `123456`, will the choice vary? Answering this question necessitates an adaptive, sematic-aware cross-site guessing model.

Fortunately, we find that TarGuess-III can fulfill this goal by introducing the PII-based tags (which we have proposed in $\mathcal{G}_I$ of TarGuess-I) into the grammar $\mathcal{G}_{II}$ of TarGuess-II. In this way, we can build a PII-enriched, password reuse-based grammar $\mathcal{G}_{III}$. More specifically, besides the L, D, S tags in $\mathcal{G}_{II}$, $\mathcal{G}_{III}$ considers six types of PII usages as with $\mathcal{G}_I$, and adds a number of *type-based* PII tags (e.g., N_1∼N_7 and B_1∼B_{10} as shown in Sec. 4.1) into $\mathcal{V}$ of $\mathcal{G}_{II}$.

In the training phase, all the PII-based password segments (each of which is parsed with one kind of PII tag) only involve the six structure-level transformation rules as defined in $\mathcal{G}_{II}$, and all the other things in $\mathcal{G}_{III}$ remain the same with that of $\mathcal{G}_{II}$. In the guess generation phase, from $\mathcal{G}_{III}$ we derive: (1) all the terminals (e.g., `love@1314`) which are instantiated from base structures that only consist of L, D and S tags; and (2) all the pre-terminals (e.g., $\mathsf{N}_4\mathsf{B}_5$ and $\mathsf{N}_3$1314) which are intermediate guesses that consist of PII-based tags. For these intermediate guesses, we further instantiate them with the target user's PII.

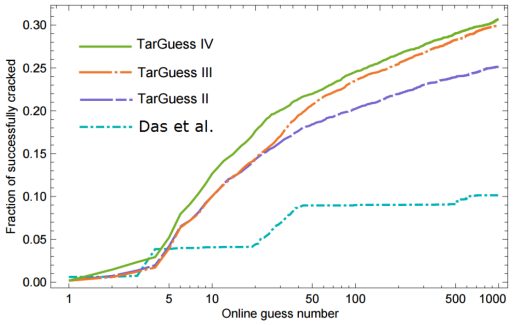


Fig. 12: A comparison of TarGuess II∼IV and Das et al.'s algorithm [11], trained on the 66,573 non-identical PW pairs of 126→CSDN and tested on the 30,8045 non-identical PW pairs of Dodonew→CSDN. Besides a sister password, TarGuess III uses four types of 51job type-1 PII and TarGuess IV further uses the gender info.

As with $\mathcal{G}_I$, our $\mathcal{G}_{III}$ is also highly adaptive. The reasons are similar with that of $\mathcal{G}_I$ (see Sec. 4.1). This means that

a new semantic tag, namely W_1 for website name, can be easily incorporated into $\mathcal{G}_{III}$ as with these PII tags. Now, $\mathcal{G}_{III}$ can parse `Alice1978Yahoo` into the structure $\mathsf{N}_4\mathsf{B}_5\mathsf{W}_1$, and the guess `Alice1978eBay` can be generated with the highest probability, because *no* transformation rules will be involved in the process from $\mathsf{N}_4\mathsf{B}_5\mathsf{W}_1$ to `Alice1978eBay`.

As shown in Fig. 12, even if U does not exactly reuse her Dodonew password for her CSDN account, TarGuess-III still can achieve a success rate of 23.48% when allowed to try only 100 guesses, being 16.3% more effective than TarGuess-II.

4.4 TarGuess-IV

TarGuess-IV aims to online guess a user U's passwords by exploiting U's one sister password as well as both type-1 and type-2 PII. A major challenge is that, type-2 PII (e.g., gender) can not be directly measured using any PCFG grammars or Markov n-grams. We tackle this issue by proving a theorem and leveraging the Bayesian theory.

THEOREM 1. *Let pw denote the event that the password `pw` is selected by U for a service, pw' denote the event that `pw`$'$ is selected by U for another service and was leaked, A_i $(i = 1, 2, \cdots, n)$ denote one kind of user PII attributes, including both type-1 and type-2 ones. We have*

$$\Pr(pw|pw', A_1, A_2, \cdots, A_n) = \frac{\prod_{i=1}^{n} \Pr(pw|pw', A_i)}{\Pr(pw|pw')^{n-1}},$$

under the assumptions that $A_1, A_2, \cdots, A_n$ are mutually independent, and that they are also mutually independent under the events pw' and (pw, pw').

Proof. Since $A_1, \cdots, A_n$ are assumed to be mutually independent, and thus we have $\Pr(A_1, \cdots, A_n) = \prod_{i=1}^{n} \Pr(A_i)$; Since they are also assumed to be mutually independent under the events pw' and (pw, pw'), and thus $\Pr(A_1, \cdots, A_n|pw) = \prod_{i=1}^{n} \Pr(A_i|pw)$, $\Pr(A_1, \cdots, A_n|pw') = \prod_{i=1}^{n} \Pr(A_i|pw')$ and $\Pr(A_1, \cdots, A_n|pw, pw') = \prod_{i=1}^{n} \Pr(A_i|pw, pw')$. Now, we can derive:

$$\begin{aligned}
&\Pr(pw|pw', A_1, A_2, \cdots, A_n)\\
=&\frac{\Pr(pw, A_1, A_2, \cdots, A_n|pw')}{\Pr(A_1, A_2, \cdots, A_n|pw')}\\
=&\frac{\Pr(A_1, A_2, \cdots, A_n|pw, pw') \cdot \Pr(pw|pw')}{\prod_{i=1}^{n} \Pr(A_i|pw')}\\
=&\frac{\left(\prod_{i=1}^{n} \Pr(A_i|pw, pw')\right) \cdot \Pr(pw|pw')}{\prod_{i=1}^{n} \Pr(A_i|pw')}\\
=&\frac{\prod_{i=1}^{n} \left(\Pr(A_i|pw, pw') \cdot \Pr(pw|pw')\right)}{\left(\prod_{i=1}^{n} \Pr(A_i|pw')\right) \cdot \Pr(pw|pw')^{n-1}}\\
=&\frac{\prod_{i=1}^{n} \frac{\Pr(pw, A_i|pw')}{\Pr(A_i|pw')}}{\Pr(pw|pw')^{n-1}} = \frac{\prod_{i=1}^{n} \Pr(pw|pw', A_i)}{\Pr(pw|pw')^{n-1}}.
\end{aligned}$$

■

Note that, the assumptions in Theorem 1 do not contradict with the fact that $A_1, \cdots, A_n$ are *dependent* on pw' and (pw, pw'). This theorem indicates that the problem of predicting a user U's `pw`$'$ at one service, when given U's PII info $A_1, A_2, \cdots, A_n$ and the sister password `pw`, can be addressed by a "divide-and-conquer" approach. More specifically, we can first compute $\Pr(pw|pw', A_i)$ for each i and $\Pr(pw|pw')$, and then compute the final goal $\Pr(pw|pw', A_1, A_2, \cdots, A_n)$. Fortunately, $\Pr(pw|pw')$ can be computed using TarGuess-II, and $\Pr(pw|pw', A_i)$ can be obtained by using TarGuess-III when A_i is a type-1 PII. The only issue left is how to compute $\Pr(pw|pw', A_i)$ when A_i is a type-2 PII.

To address it, we introduce the Bayesian theory. Without loss of generality, assume A_k is a type-2 PII. First,

$$\begin{aligned}
\Pr(pw, pw', A_k) &= \Pr(pw, A_k|pw') \cdot \Pr(pw')\\
&= \Pr[(A_k|pw)|pw'] \cdot \Pr(pw|pw') \cdot \Pr(pw').
\end{aligned}$$

It is reasonable to approximate $\Pr[(A_k|pw)|pw']$ by $\Pr(A_k|pw)$. Consequently, we have:

$$\begin{aligned}
\Pr(pw|pw', A_k) =& \frac{\Pr(pw, A_k, pw')}{\Pr(pw', A_k)}\\
=& \frac{\Pr[(A_k|pw)|pw'] \cdot \Pr(pw|pw') \cdot \Pr(pw')}{\Pr(pw', A_k)} \quad (1)\\
\approx& \Pr(pw|pw') \cdot \frac{\Pr(A_k|pw) \cdot \Pr(pw')}{\Pr(pw', A_k)},
\end{aligned}$$

where $\Pr(pw|pw')$ is called the "prior" in Bayesian theory, the factor $\Pr(A_k|pw) \cdot \Pr(pw')/\Pr(pw', A_k)$ represents the impact of (pw', A_k) on the probability of event (pw', pw).

As a result, $\Pr(pw|pw', A_k)$ can be fully computed: (1) $\Pr(pw|pw')$ can be computed using TarGuess-II; (2) $\Pr(pw', A_k) = c_1$ is a constant, because the event (pw', A_k) is a known and fixed fact when we attack U, and the ordering of guesses do not need the exact value of c_1; (3) $\Pr(pw')$=c_2 is, similarly, a constant since the event pw' is a known and fixed fact when we attack U; and (4) $\Pr(A_k|pw)$ can be computed by counting the training set—the password `pw` is selected by what fraction of users with an attribute A_k (e.g., female). Note that, when the training data is sufficiently large (e.g., in millions), $\Pr(A_k|pw)$ can be got by direct counting; Otherwise, smoothing techniques (e.g., Laplace and Good-Turing) shall be used to assure accuracy.

We note that some PII attributes are inherently *dependent* between each other (e.g., birthday vs. age, and name vs. gender). Fortunately, since most of the PII attributes (see Table 1) are largely *independent*, the practicality of Theorem 1 will not be affected much. This is especially true when many attributes are simultaneously exploited. We observe that, even if TarGuess-III only employs birthday and TarGuess-IV employs one more PII (i.e., age), TarGuess-IV still performs better than TarGuess-III by now only adjusting the *non*-birthday-involved guesses using Eq. 1.

As shown in Fig. 12, by exploiting an additional PII (i.e., gender), TarGuess-IV is able to achieve improvements over TarGuess-III in the range 4.38%∼18.19% within 10∼10^3 guesses, reaching a success rate of 24.51% with 10^2 guesses and 30.66% with 10^3 guesses. This indicates that *type-2 PII, which, as far as we know, has never been considered in the literature of password cracking, is indeed valuable for $\mathcal{A}$.*

4.5 Dealing with other attacking scenarios

As mentioned in Table 2, seven scenarios can be resulted from the various combinations of the 3 types of personal info that we focus in this work. This means that, beyond the four types of most representative scenarios #1∼#4 that we have considered above, three other ones remain: #5 (type-2 PII), #6 (type-1 PII + type-2 PII) and #7 (1 sister PW + type-2 PII). Besides, there are scenarios involving 2^+ sister PWs: #8 (2^+sister PWs) and #9 (2^+sister PWs+some PII).

When A_k is type-2 PII, it is natural to derive from Eq. 1 that:

$$\Pr(pw|A_k) \approx \Pr(pw) \cdot \frac{\Pr(A_k|pw)}{\Pr(A_k)}, \quad (2)$$

where $\Pr(A_k) = c_3$ is a constant, and both $\Pr(pw)$ and $\Pr(A_k|pw)$ can be obtained by counting the training set, as discussed in Sec. 4.4. Eq. 2 well addresses Scenarios #5.

To tackle Scenario #6, we need to develop a new formulation. From Theorem 1, we can derive that

Table 11: Training and test settings for each attacking scenario under 9 algorithms(Tra. opt.=Trawling optimal)†

Experimental scenario*	Training set(s), with policy and language consistent with the test set								Test set (size; service)
	PCFG	Markov	Tra. opt.	Personal-PCFG	TarGuess-I	Das et al.	TarGuess-II‡	TarGuess-III, IV‡	
#1: 12306→Dodo	126	126	Dodo	PII-12306	PII-12306	126	12306, 126	12306, PII-126	49,775; Dodo
#2: 12306→CSDN	8^+Dodo	8^+Dodo	CSDN	8^+PII-Dodo	8^+PII-Dodo	8^+Dodo	12306, 8^+Dodo	12306, 8^+PII-Dodo	12,635; CSDN
#3: Dodo→CSDN	8^+Dodo	8^+Dodo	CSDN	8^+PII-Dodo	8^+PII-Dodo	8^+Dodo	Dodo, 8^+126	Dodo, 8^+PII-12306	5,997; CSDN
#4: Dodo→12306	Dodo	Dodo	CSDN	PII-Dodo	PII-Dodo	Dodo	Dodo, 126	PII-Dodo, 126	49,775; 12306
#5: CSDN→12306	Dodo	Dodo	12306	PII-Dodo	PII-Dodo	Dodo	CSDN, Dodo	CSDN, PII-Dodo	12,635; 12306
#6: CSDN→Dodo	126	126	Dodo	PII-12306	PII-12306	126	CSDN, 126	CSDN, PII-126	5,997; Dodo
#7: Rootkit→Yahoo	RY	RY	Yahoo	PII-Rootkit	PII-Rootkit	RY	RY, Xato	Rock, PII-Xato	214; Yahoo
#8: Rootkit→000web	L+D RY	L+D RY	000web	L+D PII-Rootkit	L+D PII-Rootkit	L+D RY	RY, L+D Xato	RY, L+D PII-Xato	2,949; 000web
#9: 000web→Rootkit	RY	RY	Rootkit	PII-Xato	PII-Xato	RY	000web, Xato	000web, PII-Xato	2,949; Rootkit
#10: Yahoo→Rootkit	RY	RY	Rootkit	PII-Xato	PII-Xato	RY	Yahoo, Xato	Yahoo, PII-Xato	214; Rootkit

†Dodo=Dodonew; RY=Rockyou; 000web=000webhost; PII-X=PII-associated X; 8^+=$len \geq 8$; L+D=PWs with both letters and digits.
*$A \rightarrow B$ means that: (1) for the four PW-reuse-based algorithms, a user U's PW at service A can be used by $\mathcal{A}$ to help attack U's account at service B; and (2) for the other five algorithms, U's PW at A is not involved, and only U's password at B is used as the target. Note that, every user's PWs in both A and B now have been associated with PII (see Tables 12 and 13) to facilitate the four PII-based algorithms.
‡When training TarGuess-II∼IV, U's one sister PW comes from the 1st dataset, and $\mathcal{A}$ uses it to guess U's password from the 2nd dataset.

$$\Pr(pw|A_1, A_2, \cdots, A_n) = \frac{\prod_{i=1}^{n} \Pr(pw|A_i)}{\Pr(pw)^{n-1}}, \quad (3)$$

where $\Pr(pw|A_i)$ can be obtained by using TarGuess-I when A_i is a type-1 PII, or be obtained by using Eq. 2 when A_i is a type-2 PII. This addresses Scenario #6.

As our Theorem 1 is suitable for both type-1 and type-2 PII, Scenario #7 can be readily tackled by first using TarGuess-II to generate a list of guesses and then adjusting the probabilities of the guesses according to Eq.1.

For Scenarios #8 and #9, they cannot be readily addressed using the models proposed in this work, and we leave them for future work. Still, as we have shown in Sec. 2, only a marginal fraction of users have leaked two or more passwords, and thus these two scenarios are far less realistic than the 7 targeted guessing scenarios we have addressed.

Summary. We have designed a series of sound probabilistic models for targeted online guessing, with each characterizing one of the seven types of attacking scenarios. Our TarGuess-I and II significantly outperform the related algorithms [11, 17], while TarGuess-III and IV, for the first time, tackle the realistic issues of *combining* users' leaked passwords and PII to facilitate online guessing. Based on TarGuess-I∼IV, we further show how to address the three remaining scenarios. Extensive experiments in the following further demonstrate the effectiveness of our TarGuess-I∼IV.

5. EXPERIMENTS

We first describe our experimental methodologies, including the 5 pair-wised lists used and the choices of various training vs. testing scenarios. Then, we provide a comparative evaluation of TarGuess-I∼IV with 5 leading algorithms.

5.1 Experiment setup

Among the nine algorithms to be evaluated, three (i.e., Markov [18], PCFG [27] and Trawling optimal [5]) only need some training passwords, four (i.e., Das et al.'s algorithm [11], TarGuess-II∼IV) work on password pairs of the same user, and four (i.e., Personal-PCFG [17], TarGuess-I, III and IV) involve various types of user personal info. However, only two of our original datasets (i.e., 12306 and Rootkit) are with PII (see Table 3). Thus, we build two new PII-associated PW lists, one by matching Dodonew with 51job and 12306 using email address, another similarly by matching 000webhost with Rootkit. This results in four PW lists with PII, and for their detailed number of elements, see the first row in Tables 12 and 13. Matching by email ensures that *all our PII-associated English passwords are created by Rootkit hackers, who well represent security-savvy users.* Since the Rockyou dataset is without email or user name, we further match Xato with Rootkit to obtain 15,304 PII-associated Xato passwords to supplement Rockyou.

As shown in Table 12, we further build three lists of *password pairs* for Chinese users by matching Dodonew and CSDN with the two PII-associated Chinese password lists using email address. For instance, the list Dodonew↔12306 has a total of 49,775 password pairs, of which 14,380 pairs are with non-identical passwords. Similarly, we build three lists of password pairs for English users (see Table 13), but eliminate one of them (i.e., Yahoo↔000webhost) because the limited size of test set (i.e., 96) would make it impossible to reflect the true nature of an algorithm. These five pair-wised lists lead to ten experimental scenarios in Table 11.

Table 12: Basic info of the matched Chinese datasets

Original dataset	12306(129,303 with PII)		Dodonew(161,510 with PII)	
	Total	Non-identical(%)	Total	Non-identical(%)
Dodonew	49,775	14,380 (28.89%)	—	—
CSDN	12,635	5,538 (43.83%)	5,997	3,957(65.98%)

Table 13: Basic info of the matched English datasets

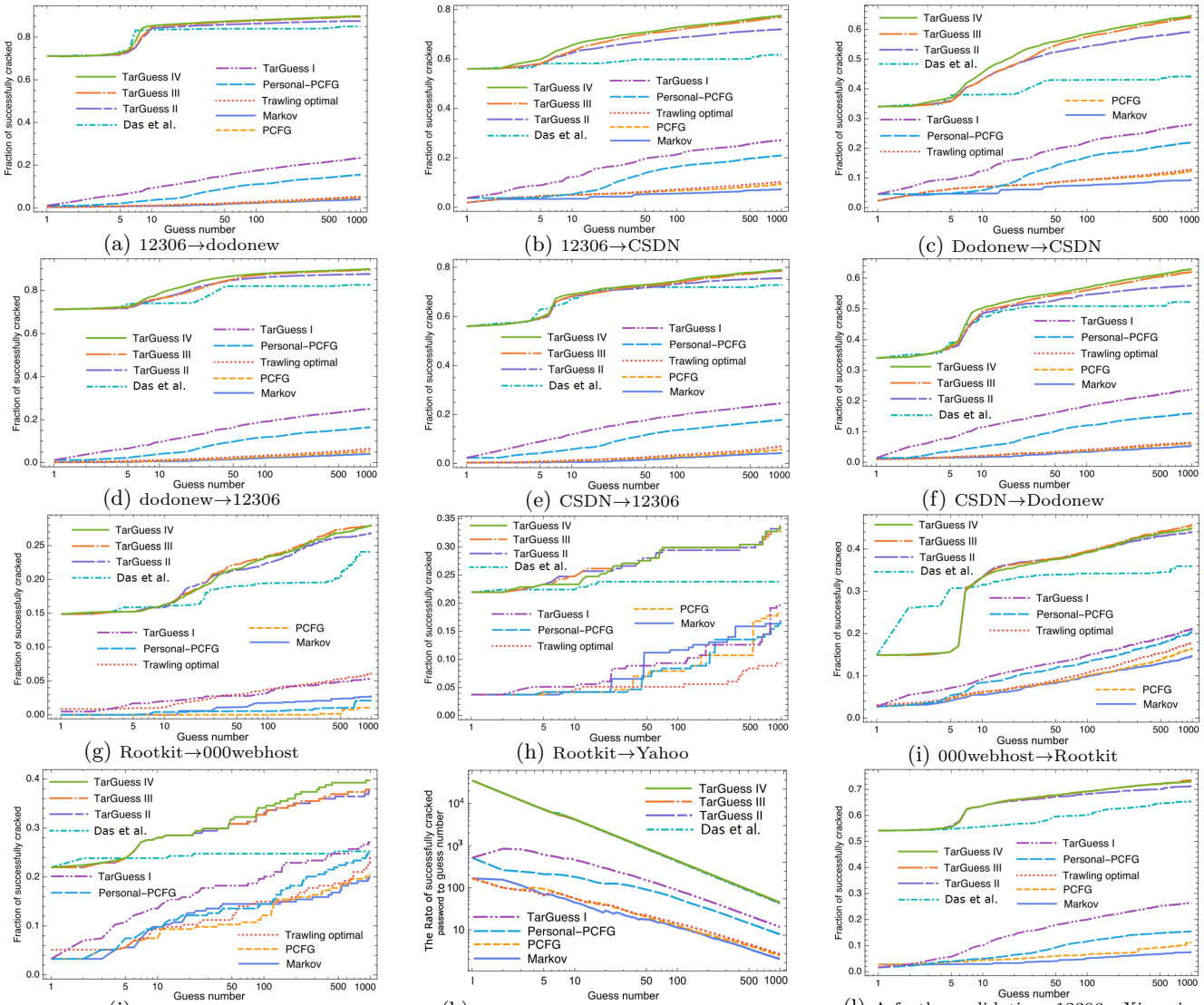
Original dataset	Rootkit(69,330 with PII)		000webhost(2,950 with PII)	
	Total	Non-identical(%)	Total	Non-identical(%)
000webhost	2,949	2,510 (85.11%)	—	—
Yahoo	214	167 (79.04%)	96	90 (93.75%)

To make our experiments as realistic as possible, our choices of the training set(s) for a given test set (attacking scenario) adhere to three rules: (1) they never come from the same service; (2) they are of the same language and password policy; and (3) the training set(s) shall be as large as possible. Rule 1 prevents our experiments from the overfitting issue, while rules 2 and 3 ensure the effectiveness of each algorithm. For fair comparison, we further make sure that all the 9 algorithms work on the same test set, and that for the same type of algorithms (e.g., TarGuess-I and [17]), their training sets exploit the same personal info.

5.2 Evaluation results

The guess number is the most scarce resource when performing an online attack, while computational power and bandwith are not essential. For instance, in every of these ten experiments, the training phase can be completed on a common PC in less than 65.3s, while the generation of 1000 guesses for each user takes less than 2.1s. Thus, we use the guess-number-graph to evaluate the effectiveness of our four probabilistic algorithms with five leading algorithms (i.e., PCFG [27], Markov [18], Trawling optimal [5], Personal-PCFG [17] and Das et al.'s cross-site algorithm [11]).

Figs. 13(a)∼13(j) show that, under the same personal info available to $\mathcal{A}$ and within 100 guesses: (1) TarGuess-I drastically outperforms its foremost counterpart, Personal-PCFG [17], by 11.17%∼ 509% (avg. 84%); (2) TarGuess-II drastically outperforms Das et al.'s algorithm [11] by 8.12%∼300% (avg. 111.06%) when cracking non-identical password pairs; and (3) TarGuess-III and IV are able to gain a success rate of 73.09% when attacking Chinese common users and of 31.61% when attacking English security-savvy users. As the guess number increases, in most cases the superiorities of TarGuess-I∼IV over their counterparts are



(a) 12306→dodonew (b) 12306→CSDN (c) Dodonew→CSDN

(d) dodonew→12306 (e) CSDN→12306 (f) CSDN→Dodonew

(g) Rootkit→000webhost (h) Rootkit→Yahoo (i) 000webhost→Rootkit

(j) Yahoo→Rootkit (k) Diminish returns in all algorithms (l) A further validation: 12306→Xiaomi

Fig. 13: Experiment results for 11 targeted guessing scenarios. Sub-figures (a) to (f) represent attacks on common users, while sub-figures (g) to (j) represent attacks on security-savvy users. Our TarGuess-I∼IV are highly effective.

further enhanced. Here we mainly focus on cracking efficiency of cross-cite algorithms for non-identical PW pairs, because cracking non-identical pairs is their primary goal.

Particularly, TarGuess-IV, for the first time, characterize a very powerful yet realistic attacker who can launch cross-site guessing by exploiting a victim's one sister password as well as both type-1 and 2 PII. As shown in Figs. 13(a) and 13(f), within 10 guesses, TarGuess-IV can gain success rates 45.49%∼85.33% (avg. 65.70%) against accounts of common users at various services; Within 10^2 guesses, the figures are 56.96%∼88.02% (avg. 73.08%); Within 10^3 guesses, the figures are 62.95%∼89.87% (avg. 77.32%). To achieve such success rates, state-of-the-art trawling guessing algorithms [18, 24] would need 10^{13} guesses per user and take several days by using high performance computers.

We discover that password strength against both targeted guessing and trawling guessing follows *a power law distribution*. The first few guesses are extremely effective, e.g., at 10 guesses, TarGuess-I can gain a chance over 9.25% against every Chinese service. Yet, as the guess number increases, the success rate of each attempt decreases rapidly. Interestingly, in most of our experiments, we find that for every of the eight real-world algorithms (i.e., excluding the trawling optimal one [5]), the ratio of *the number of successfully cracked accounts* f_n to *the guess number per account* n can be well approximated by a power law: $f_n = C * n^s$, where s and C are constants dependent on the test set. As shown in Fig. 13(k), such a relationship is a straight line when plotted on a log-log style. Moreover, the three parallel layers in Fig. 13(k) just correspond to three kinds of guessing algorithms: trawling, targeted using PII, targeted using a sister PW. This diminishing principle has an important implication: $\mathcal{A}$ would stop at some point as her gains do not weigh her costs, and there would be three such points corresponding to three different attacking strategies, yet existing guidelines [14, 16] only consider the trawling point.

Our results suggest that the majority of normal users' passwords are vulnerable to 100 targeted online guesses (e.g., as allowed by NIST for level 1 and 2 systems [7]), invalidating the claim that online guessing "can be readily addressed by throttling the rate of login attempts permitted" [16]. Therefore, our targeted online guessing is a much more damaging threat to password security than trawling guessing and than the password community (see [14, 16]) might have expected. Our models would facilitate better evaluation of the practicality of existing and future password policies (see [8, 26]), and they would also be very helpful for forensic investigators to recover passwords in an offline manager.

5.3 A further validation

In the above experiments, our test sets are in plain-text. Would our algorithms be still effective when cracking "real accounts" about which little is known? We confirm this with a further experiment to crack Xiaomi passwords, which are MD5 hashed with salting, leaked from the world's 3rd largest smartphone maker (ranked after Apple and Samsung).

We attack Xiaomi as we crack Dodonew in Scenario #2 of Table 11. The test set contains 5,284 Xiaomi hashes which are obtained after matching the 8M Xiaomi dataset with the 130K 12306 dataset using email. As shown in Fig. 13(l), within $10\sim10^3$ guesses, TarGuess-I outperforms Personal-PCFG [17] by 70.58%$\sim$ 119%; TarGuess-II outperforms Das et al.'s algorithm [11] by 73.66%$\sim$ 405%; TarGuess-III and IV can gain a success rate of 63.61%$\sim$73.56%. These results well accord with our above 10 experiments, especially the Chinese ones, suggesting the generality of our models.

6. CONCLUSION

We have presented the first systematic evaluation of the extent to which an online password guessing attacker can gain advantages by exploiting various types of user personal info including leaked passwords and common PII. Our study is grounded on a framework that consists of 7 sound probabilistic models, with each dealing with one typical attacking scenario. Particularly, TarGuess-I$\sim$IV characterize the four most representative scenarios, and for the first time, the problem of how to model context-aware, semantic-enriched cross-site password guessing attacks has been well addressed.

Extensive experimental results show that TarGuess-I and II drastically outperform their foremost counterparts, while TarGuess-III and IV can gain success rates as high as 73% with just 100 guesses. Our results suggest that, the currently used security mechanisms would be largely ineffective against the targeted online guessing threat, and this threat has already become much more damaging than expected. We believe that the new algorithms and knowledge of effectiveness of targeted guessing models can shed light on both existing password practice and future password research.

7. REFERENCES

[1] *All Data Breach Sources*, May 2016. https://breachalarm.com/all-sources.

[2] *Turkey: personal data of 50 million citizens leaked online*, April 2016. http://bit.ly/1TPA4j4.

[3] *Amid Widespread Data Breaches in China*, Dec. 2011. http://www.techinasia.com/alipay-hack/.

[4] D. V. Bailey, M. Dürmuth, and C. Paar. Statistics on password re-use and adaptive strength for financial accounts. In *Proc. SCN 2014*, pages 218–235.

[5] J. Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proc. IEEE S&P 2012*, pages 538–552.

[6] J. Bonneau, C. Herley, P. van Oorschot, and F. Stajano. Passwords and the evolution of imperfect authentication. *Commun. ACM*, 58(7):78–87, 2015.

[7] W. Burr, D. Dodson, R. Perlner, S. Gupta, and E. Nabbus. NIST SP800-63-2: Electronic authentication guideline. Technical report, NIST, Reston, VA, Aug. 2013.

[8] X. Carnavalet and M. Mannan. A large-scale evaluation of high-impact password strength meters. *ACM Trans. Inform. Syst. Secur.*, 18(1):1–32, 2015.

[9] A. Chaabane, G. Acs, M. A. Kaafar, et al. You are what you like! information leakage through users' interests. In *Proc. NDSS 2012*, pages 1–15.

[10] C. Custer. *China's Internet users zoom to 668 million*, Jan. 2016. http://www.apira.org/news.php?id=1736.

[11] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The tangled web of password reuse. In *Proc. NDSS 2014*, pages 1–15.

[12] M. Dell'Amico and M. Filippone. Monte carlo strength evaluation: Fast and reliable password checking. In *Proc. ACM CCS 2015*, pages 158–169.

[13] M. Dürmuth, D. Freeman, and B. Biggio. Who are you? A statistical approach to measuring user authenticity. In *Proc. NDSS 2016*, pages 1–15.

[14] D. Florêncio, C. Herley, and P. van Oorschot. An administratorąś guide to internet password research. In *Proc. USENIX LISA 2014*, pages 44–61.

[15] T. Fox-Brewster. *13 Million Passwords Appear To Have Leaked From 000webhost*, Oct. 2015. http://t.cn/R4tKrEU.

[16] P. A. Grassi and J. L. Fenton. NIST SP800-63B: Digital authentication guideline. Technical report, NIST, Reston, VA, 2016.

[17] Y. Li, H. Wang, and K. Sun. A study of personal information in human-chosen passwords and its security implications. In *INFOCOM 2016*, pages 1–9.

[18] J. Ma, W. Yang, M. Luo, and N. Li. A study of probabilistic password models. In *Proc. IEEE S&P 2014*, pages 689–704.

[19] M. L. Mazurek, S. Komanduri, T. Vidas, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur. Measuring password guessability for an entire university. In *Proc. ACM CCS 2013*, pages 173–186.

[20] E. McCallister, T. Grance, and K. Scarfone. NIST SP800-122: Guide to protecting the confidentiality of personally identifiable information (PII). Technical report, NIST, Reston, VA, April, 2010.

[21] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proc. ACM CCS 2005*, pages 364–372.

[22] *Four Years Later, Anthem Breached Again: Hackers Stole Credentials*, Feb. 2015. http://t.cn/RqWrMKC.

[23] *Senate Bill No. 1386: Personal information*, Sep. 2002. http://bit.ly/1WJIIpK.

[24] B. Ur, S. M. Segreti, L. Bauer, and et al. Measuring real-world accuracies and biases in modeling password guessability. In *USENIX SEC 2015*, pages 463–481.

[25] D. Wang, D. He, H. Cheng, and P. Wang. fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars. In *Proc. IEEE/IFIP DSN 2016*, pages 595–606.

[26] D. Wang and P. Wang. The emperor's new password creation policies. In *ESORICS 2015*, pages 456–477.

[27] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In *Proc. IEEE S&P 2009*, pages 391–405.

[28] *Nearly 80 percent of Internet users suffer identity leaks*, July 2015. http://www.chinadaily.com.cn/china/2015-07/24/content_21400381.htm.

[29] Y. Zhang, F. Monrose, and M. Reiter. The security of modern password expiration:an algorithmic framework and empirical analysis. In *Proc. ACM CCS 2010*, pages 176–186.