A DIVERSITY-BASED FRAMEWORK FOR DYNAMIC PASSWORD POLICY GENERATION

A Thesis
Presented to
The Academic Faculty

by

Shukun Yang

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2016

A DIVERSITY-BASED FRAMEWORK FOR DYNAMIC PASSWORD POLICY GENERATION

Approved by:

Professor Raheem Beyah, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor John Copeland
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henry Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 1 April 2016

To my parents Leijia Xin and Huacong Yang,

without whom none of my achievements would be possible.

Thank you for your unconditional love and support.

ACKNOWLEDGEMENTS

Foremost, I want to thank my advisor, Professor Raheem Beyah, for constant guidance, support and trust. His ideas and suggestions have significantly inspired and encouraged me over the entire course of my graduate research. He has also taught me various soft skills that I would benefit from for life. I am fortunate to have worked with such a great advisor.

I would like to thank Professor John Copeland and Professor Henry Owen for serving as my thesis reading committee. I also want to thank all the members in the Communications Assurance and Performance (CAP) Group, especially Shouling Ji, Weiqing Li, Xiaojing Liao, and Qinchen Gu, who have been there for me whenever I encountered roadblocks. Last but not least, I want to thank my family and friends for their continuous support, understanding, and company.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

To keep password users from creating simple and common passwords, major websites and applications provide a password-strength measure, namely a password checker that displays instant password strength ratings in levels e.g., "strong", "moderate", and "weak". While critical requirements for a password checker to be stringent have prevailed in the study of password security, we find that regardless of the stringency, such static checkers can leak information and actually help the adversary enhance the performance of their attacks. To address this weakness, we propose and devise the *Dynamic Password Policy Generator*, namely *DPPG*, to be an effective and usable alternative to the existing password strength checker. *DPPG* aims to enforce an evenly-distributed password space and generate dynamic policies for users to create passwords that are diverse and contribute to the overall security of the password database. Since *DPPG* is modular and can function with different underlying metrics for policy generation, we further introduce a diversity-based password security metric that evaluates the security of a password database in terms of password space and distribution. The metric is useful as a countermeasure to well-crafted offline cracking algorithms and theoretically illustrates why *DPPG* works well.

CHAPTER I

INTRODUCTION

Text-based passwords have been used widely in both online and offline applications for decades. Since passwords are personal and portable, they are not likely to be replaced in the foreseeable future [49]. However, the phenomenon that people choose simple passwords and reuse common passwords [38] has raised great security concerns as such passwords are vulnerable to offline cracking attacks. To make things worse, a number of password leakage incidents [5, 7, 10, 3] have happened recently and frequently. Large datasets of leaked passwords can greatly enhance the attackers' capability in conducting training-based password attacks, thus posing significant threats on password security.

The most direct and pervasive protective mechanism used by major websites and applications is the password strength checker [39], which evaluates the strength of passwords proactively during user registration. While the goal is to guide users to create strong passwords, in previous work [108, 23, 55], the lack of accuracy and consistency in the strength feedback has been widely observed and examined. That is, existing checkers do not demonstrate effective or uniform characterization of strong passwords. Furthermore, the space for the rules and policies of the checkers to be stringent is very limited as researchers have shown that the complexity of a password is a trade-off with the usability [113]. Therefore, password strength checkers simply cannot demand users to create passwords that are too complex.

On the other hand, the password strength checker itself can be a vulnerability, which has not been studied in previous research. By defining a set of password creation policies and showing users password strength scores, password checkers can exert

a strong bias on the password characteristics, especially when the policies and scoring mechanisms remain static. The passwords registered to a database are largely similar to the specific password patterns enforced by the associated checker. Although password checkers vary among websites, they inevitably rely on similar rules that focus on specific password properties (e.g., length, number of digits and special characters). When rules are relatively relaxed, password users may create simple passwords following a common distribution. When rules are relatively demanding, the password distribution is closely correlated to the scoring metrics and can be inferred. Since the password checkers are publicly available, attackers can easily make use of the password checkers to learn the password characteristics distribution that is shaped by the password checkers.

Our main contributions in this thesis are summarized as follows.

- In Chapter 3, we evaluate the impact of misusing current commercial password strength checkers from the attacker's perspective and explore the possibility and potential to leverage the checkers in offline cracking attacks. Using an attack-based model, we show that the password checkers are effective for attackers to facilitate password cracking. With a certain amount of computational power, the attacker can compromise more passwords with a specific rating with the help of the strength checkers. This implies that the static policies and scoring mechanisms used by the password strength checkers exert bias on the password characteristics distribution. Passwords with the same rating follow an obvious pattern which can be exploited by the attacker to refine the training data.

- In Chapter 4, to propose a countermeasure to protect the information on password distribution and to reduce the efficacy of well-crafted training-based attacks, we devise the *Dynamic Password Policy Generator*, namely *DPPG*, which generates dynamic password policies for users. Each new user obtains a different password policy to follow, which is generated in real-time from the server based

on but not reflecting the current password distribution. The policies thus are not static and a user does not know what policies others receive. *DPPG* works to even out the password distribution in the database and expand the password space. Since the policies users receive are dynamic, and unpredictable, the adversary cannot use them to infer the characteristics of the password database or select password training data.

- To further understand the password distribution and evaluate the threats posed by the exposure of the password distribution of a leaked dataset, we introduce the concept of *password diversity* and propose a diversity-based metric to measure the security of a password dataset in Chapter 5. The metric considers an aggregation of password properties to analyze the password characteristics distribution within a dataset. It assigns a higher security score to a password dataset with a more uniform password distribution. The metric serves as the underlying mechanism used in *DPPG* to generate dynamic policies and aims to minimize biased password distributions and to expand the usable password space. Since the training-based attacks become powerful due to strong similarities between the training and target passwords, it is meaningful to study such a metric.

CHAPTER II

LITERATURE REVIEW

In this chapter, we clarify the terminology used throughout the thesis and summarize the related work in major domains of password security research.

2.1 Terminology

Within the domain of password cracking research, we use **training-free** to refer to cracking algorithms that do not need input passwords data and generate guesses independently. **Training-free** algorithms can quickly compromise simple and common passwords, e.g., dictionary words. We use **training-based** on the other hand to denote algorithms that require training data, which work better to crack more complicated passwords with human textual patterns. When studying similarity, we use password **characteristics**, **attributes**, and **properties** interchangeably to refer to password length, number of character types, structure, and etc. We use **password space** to denote the variability of passwords with a set of constraints e.g., for passwords that are 10 characters long and contain only numerical digits, the password space has two dimensions with a size of 10^{10}. We define **password distribution** in terms of password attributes in a dataset, such as the appearance frequencies of certain characters, character types and etc. In Chapter 5, we detect communities of passwords based on their similarity and **password distribution** can be approximated by the distribution of communities.

2.2 Password Cracking

John the Ripper is a popular commercial offline password cracking application and has been widely used in password security research. It has different modes such as

Single, *Wordlist*, *Incremental*, and *Markov mode*. *Single mode* performs mangling on auxiliary information, typically the usernames associated with the passwords; *Wordlist mode* is an efficient dictionary attack with optional mangling rules; and *Incremental mode* uses smart brute-forcing and attempt to exhaust the entire password space given enough time. The only training-based algorithm lies in the *Markov mode*, which trains a given password dataset to extract characteristic information and build Markov chains to generate password guesses. As shown in the previous literature [19, 55], training-based algorithms, in most scenarios, have much better cracking performance than training-free algorithms. In this thesis, we only consider the *Markov mode* and denote it as JtR.

In [109], Weir et al. proposed a training-based password cracking algorithm using Probabilistic Context-Free Grammars, namely PCFG. The algorithm first surveys the structural distribution from a training dataset. The password structure is generated by replacing each character in the passwords with its character type. In PCFG, 3 character types are considered which are, L that stands for lower-case or upper-case alphabet, D that stands for numerical digit, and S that stands for special character. In this thesis, we further use L and U to represent lower-case and upper-case alphabets separately. For example, "Pwd@1" has a structure of "ULLSD". By using the structural distribution, PCFG then applies mangling rules to the structures in the decreasing order of likelihood and generates guesses.

Taking another approach, in [75], Narayanan and Shmatikov proposed to use *standard Markov modeling techniques* to drastically reduce the password search space and designed the first Markov model-based algorithm. Markov-based cracking algorithm extracts information from the training dataset to construct Markov Chains which can enumerate the password search space efficiently based on the prior knowledge. In [24], Castelluccia et al. improved the Markov model and proposed to build an n-gram based Markov model to generate password guesses, However, the algorithm in [75, 24]

cannot generate password guesses in the decreasing order of likelihood. To address this limitation, Dürmuth et al. further proposed an improved Markov-based password cracking algorithm, namely *Ordered Markov ENumerator* in [34] which makes guesses in the decreasing order of likelihood. We denote it as OMEN for short. Furthermore, they also extended OMEN to OMEN+ and consider users' social profiles during password cracking.

2.3 Password Strength Measurement

The trade off between usability and stringency of the password requirements has also been explored extensively. In [89], Shay et al. found that users struggle with new and complex password requirements. In [71], Mazurek et al. measured the strength of passwords created by over 25K university students, faculty, and staff. They found that users who expressed annoyance with complex password creation policies tend to create more vulnerable passwords. Reporting the same issue, Shay et al. in [89] conducted a survey of 470 CMU computer users to find that users struggle with new password requirements, especially the complex ones. These work report the issues that while complex password creation policies might enhance the security of passwords, general usability is compensated. Therefore it is meaningful to ensure the proper level of complexity in password policies. In [66], Li et al. conduct an empirical analysis of Chinese web passwords. According to their statistical results, user-chosen passwords have explicit regional differences. In [17], Bonneau analyzed an anonymized corpus of 70M Yahoo! passwords. He estimated that passwords provide fewer than 10 bits of security against an online, trawling attack, and only about 20 bits of security against an optimal offline dictionary attack. He also found that graphical feedback during the password selection process makes little difference in improving password security, and seemingly distant language communities choose the same weak passwords.

Traditional password strength metric has been found ineffective through previous

work. In [108], Weir et al. evaluated various traditional metric, e.g., NIST entropy, for password creation policies by attacking leaked passwords using their PCFG based cracking algorithm. They found that the NIST entropy and other conventional metric are not effective for password security, and proposed new PCFG cracking-based password creation policies. Another work employing the password cracking idea to measure password strength is [59], where Kelley et al. analyzed 12K passwords collected under seven composition policies via an online study. They also concluded that the effectiveness of using entropy as a measure of password guessability is very limited. In [70], Ma et al. conducted a study of probabilistic password models. They proposed a *probability-threshold graph model* to capture the probability threshold distribution in log scale versus the percentage of passwords above the threshold.

Although interesting proposals have been made to replace traditional metric, the new approaches are still based on the complexity of individual passwords without considering the overall password distribution of a password database. Similarity between individual passwords and how it could affect the security of passwords in bulk are not studied.

The existing password checkers have also been studied in a variety of ways, but mainly in terms of their accuracies and effectiveness in indicating the strength of individual passwords. In [23], Carnavalet and Mannan studied existing password checkers by analyzing feedback from 11 commercial checkers on passwords in various datasets. They found significant inconsistencies among different checkers, which may confuse users. Ji et al. in [55] further conducted attack-based analysis on commercial checkers to find that many of them provide inaccurate and misleading feedback. Ur et al., in [104], also studied the effect of strength checkers on password creation. They found that password resistance could only increase when the checkers score passwords stringently. To suggest a different approach than traditional checkers, Castelluccia et al. presented *adaptive password strength meter* that estimate password strength

using Markov models [24]. They also proposed a secure implementation of the presented concept. However, the checker solely relies on n-gram which does not consider structural and other information. In [53], Houshmand and Aggarwal proposed a tool, named *Analyzer and Modifier for Passwords* (AMP), to help users choose stronger passwords. AMP first estimates a password's crackability based on the PCFG cracking model, and then modifies the weak password slightly to meet the security requirement. Komanduri et al. implemented another tool, namely *Telepathwords*, to help users create strong passwords [62]. As a user creates a password, Telepathwords prevents weak passwords by making realtime predictions of the next character that the user will type. In [42], Forget et al. also developed a tool, namely *Persuasive Text Passwords* (PTP), which leverages the persuasive technology principle to influence users in creating more secure passwords without sacrificing usability. Schmidt and Jaeger evaluated the security of automated strengthening of passwords [87]. They found that passwords that were strengthened are still susceptible to modern cracks, provided that the adversary knows the strengthening algorithm. The work most related to this thesis is [83], where Schechter et al. proposed to prevent users from creating popular passwords using a bloom filter. However, the filter only recognizes popular passwords rather than having the capability to identify popular password patterns.

Again, extensive studies have shown that password checkers lack the accuracy in reflecting the proper strength levels of passwords, and new strength checkers adopting more rigorous algorithms are observed to have better effectiveness. However, not much focus has been put on protecting the security of password distributional information. According to Kerckhoffs's principle, an attacker can learn the system and optimize the attack. In this thesis, we set aside evaluating the effectiveness of password checkers, and study the possibility of such checkers leaking crucial passwords distributional information. We also propose a new mechanism to render dynamic policies that

minimize the attacker's ability to learn the system.

2.4 Password Security Evaluation

In [38], Florêncio and Herley conducted a large scale study of password habits. Several interesting facts are found in the study such as on average a user has 6.5 passwords, and each of which is shared across 3.9 different sites. Similar to [38], Gaw and Felten studied the password reuse phenomenon [45]. Based on a study of 49 undergraduate students, they concluded that the majority of users have three or fewer passwords and their passwords are reused twice. Stobert and Biddle also studied user behavior in managing multiple passwords [93]. They found that many users reuse and write down passwords. In [19], Bonneau et al. evaluated two decades of text-password alternatives. They found that many alternatives fail to consider a sufficiently wide range of real world constraints, and thus text-passwords will still be the dominating authentication method in the foreseeable future. Schmidt and Jaeger evaluated the security of automated strengthening of passwords [87]. They found that passwords that were strengthened are still susceptible to modern cracks, provided that the adversary knows the strengthening algorithm.

As summarized above, the password reuse phenomenon is pervasive and concerning. This means that a user who reuses a password frequently will have the same password in many password databases. If the number of such users are large, different password databases will resemble each other thus sharing a common password distribution. Therefore, it is meaningful to study how the password distribution will affect the security of password databases.

In [36], Fahl et al. presented a study on the ecological validity of password studies, which can help researchers design proper user study settings. In [47], Haque et al. investigated the issue of user comfort from the viewpoint of psychometrics by developing a questionnaire. Kuo et al. evaluated human selection of mnemonic

phrase-based passwords [65]. They concluded that user-generated mnemonic passwords are slightly more resistant. In [20], Bonneau and Schechter challenged the conventional wisdom that users cannot remember cryptographically-strong secrets. Through a study, they demonstrated that users can learn randomly-assigned 56-bit codes. In [27], Chiasson et al. presented a usability study of two recent password managers *PwdHash* and *Password Multiplier*. Their findings suggest that ordinary users would be reluctant to use these managers. Jeyaraman and Topkara proposed and evaluated an automatic memorable mnemonics generation system for a given password based on a text-corpus [54]. Their results show that automatic mnemonic generation is a promising technique to improve the usability of text-password systems. In [88], Shay et al. evaluated the usability of system-assigned passphrases. They found that system-assigned passphrases and passwords have similar entropy with respect to the examined usability metric.

2.5 Password Management

In [117], Yee and Sitaker designed *Passpet* for password management. In [16], Bojinov et al. introduced *Kamouflage*, a new architecture to build theft-resistant password managers. Florêncio et al. studied how to manage a portfolio of passwords in [41]. Molloy and Li evaluated the security of GridCode, a one-time password system [74]. In [44], Gasti and Rasmussen studied the security of password manager database formats. Another recent study on password managers is [67], where Li et al. found that it is still a challenge for existing password managers to be secure. In [91], Silver et al. studied the security of popular password managers.

2.6 Password Protection

In [57], Juels and Rivest proposed *Honeywords* to improve the security of hashed passwords. Florêncio and Herley studied how to securely enter passwords on a spyware infected machine in [37]. They proposed three countermeasures using passwords

embedding in random keystrokes and a shared-secret proxy. In [64], Kumar et al. proposed to reduce shoulder-surfing by using gaze-based password entry. In [48], Hart presented a phishing resistant password ceremony, *PhorceField*. To protect user accounts from password database leaks, Kontaxis et al. proposed *SAuth* [63]. In [68], Liu et al. designed and implemented *ScreenPass*, which significantly improves the security of passwords on touchscreen devices. For improving users' password management, Kim et al. introduced *YourPassword*. In [95], Stock and Johns proposed an alternative password manager design. In [28], Cox et al. presented *SpanDex* for secure password tracking in Android. In [72], McCarney et al. proposed *Tapas*, an authentication approach offering encrypted storage of passwords and theft-resistance.

CHAPTER III

COMMERCIAL PASSWORD CHECKERS

Most of the existing research only evaluates the effectiveness and helpfulness of the password strength checkers. The fact that the checkers are based on unchanged policies which indirectly bias the password characteristics distribution has not been studied. Furthermore, due to the exposure of the policies and scoring mechanisms [23, 55, 56], careful attackers can utilize the password checkers to mount more powerful attacks on passwords with high strength ratings.

3.1 Datasets, Checkers, and Crackers

Table 1 lists the 5 datasets that add up to around 81 million passwords. The datasets are leaked from several incidents [29, 70] where attackers acquire passwords by online attacking techniques. Although the password data were leaked illegally, it has been once made publicly available and used widely in password research for benevolent purposes. In our study, we use the passwords for research only without attempting to verify them.

To obtain a collection of usable password strength checkers and cracking algorithms, we conduct our experiments with PARS [55]. Bloomberg is a popular English business and news forum and QQ is a well-known Chinese portal providing numerous web services. According to evaluations in [55, 56], they provide relatively accurate

Table 1: Datasets.

Name	Size	Language	Site	Type
`Renren`	4.7M	Chinese	renren.com/	social networks
`LinkedIn`	5.4M	English	linkedin.com/	professional networks
`Tianya`	31M	Chinese	tianya.cn/	Internet forum
`Rockyou`	32.6M	English	rockyou.com/	game
`Gamigo`	6.3M	German	en.gamigo.com/	game

Table 2: Percentage of "Strong" Passwords.

checker	Gamigo	Renren	LinkedIn	Rockyou	Tianya
Bloomberg-Train	0.05%	6.30%	0.31%	0.72%	0.44%
Bloomberg-Test	0.05%	6.27%	0.31%	0.72%	0.44%
QQ-Train	12.44%	22.20%	1.75%	2.56%	5.20%
QQ-Test	12.44%	22.12%	1.74%	2.56%	5.20%

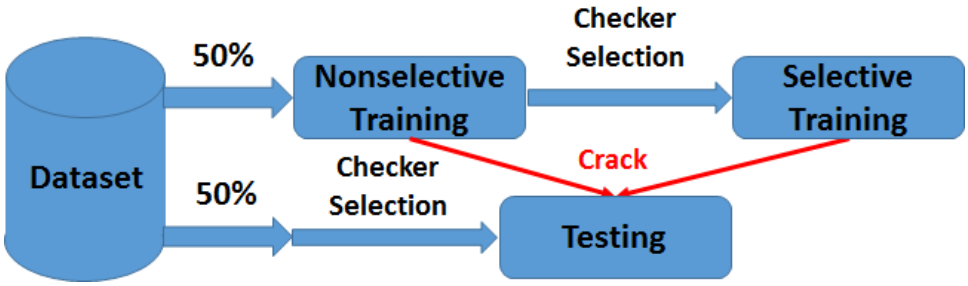


Figure 1: Attack-based Evaluation Model

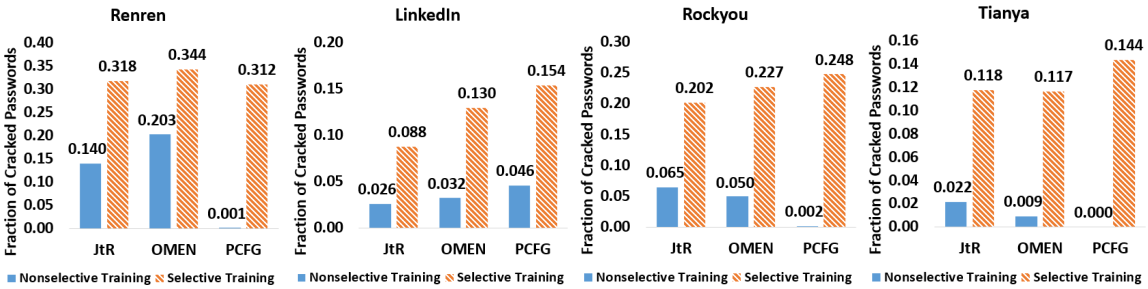
Table 3: Cross-Site Password Cracking (Bloomberg's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	-	-	-	-	-	-	3.57%	7.17%	2.59%	7.17%	2.43%	10.97%	1.75%	15.32%	1.13%	19.22%	0.29%	11.34%
LinkedIn	0.23%	1.94%	0.05%	1.14%	0.01%	10.49%	-	-	-	-	-	-	0.66%	5.87%	0.41%	7.98%	0.03%	14.29%
Rockyou	1.09%	6.90%	0.26%	4.30%	0.08%	18.59%	10.00%	17.37%	6.22%	15.54%	6.91%	21.57%	-	-	-	-	-	-
Tianya	1.43%	4.81%	0.73%	4.78%	0.01%	9.77%	2.83%	5.46%	2.82%	6.70%	1.87%	11.89%	1.14%	5.41%	1.00%	6.93%	0.16%	11.28%
Gamigo	0.67%	4.37%	0.36%	3.46%	0.00%	20.41%	4.74%	12.76%	4.80%	15.13%	6.62%	24.30%	2.13%	11.48%	1.15%	15.37%	0.24%	25.15%

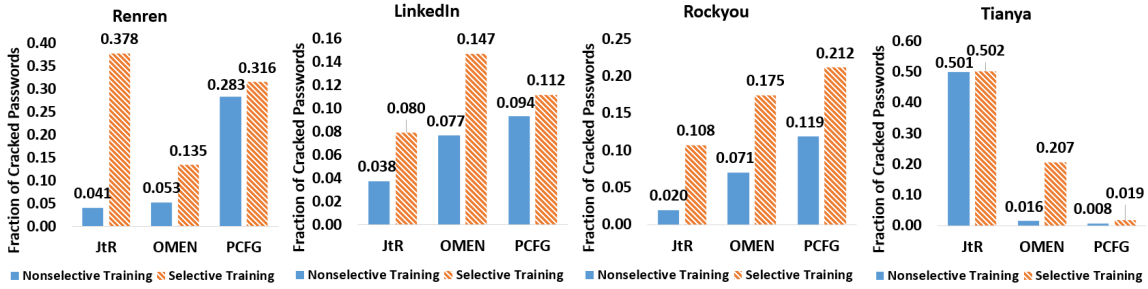
Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	1.69%	16.21%	0.80%	16.31%	0.07%	9.24%	0.15%	6.00%	0.01%	1.58%	0.12%	7.74%
LinkedIn	0.17%	3.24%	0.05%	0.85%	0.01%	9.04%	0.03%	6.48%	0.01%	1.17%	0.12%	11.47%
Rockyou	0.86%	8.84%	0.12%	1.85%	0.06%	10.79%	0.57%	15.53%	0.01%	2.70%	0.32%	19.96%
Tianya	-	-	-	-	-	-	0.07%	4.53%	0.02%	1.36%	0.07%	5.75%
Gamigo	0.55%	5.65%	0.06%	1.94%	0.00%	16.28%	0.18%	13.00%	0.00%	3.77%	0.06%	22.24%

and consistent feedback to users. There are 4 levels of password strength in both password checkers to make them comparable, and the highest rating is "strong" in common.

We use three state-of-the-art password cracking algorithms, JtR (John the Ripper-Markov) [9], OMEN (Ordered Markov ENumerator) [34], and PCFG (Probabilistic Context-free Grammar) [109], which have relatively optimal performance in password cracking as shown consistently in previous research literature.



(a) Bloomberg



(b) QQ

Figure 2: Intra-site Password Cracking (Bloomberg and QQ Password Checkers).

3.2 Threat Model: Take Your Checker, Crack Your Passwords

From an attacker's perspective, we evaluate quantitatively how existing commercial password checkers can be used to enhance offline password attacks. We are particularly interested in the pool of "strong" passwords because intuitively users trust the strength feedback and create passwords that have better ratings.

In our threat model, we assume an attacker aims to crack a target set of password hashes leaked from a website which uses a password strength checker. This means that the hashed passwords can have different strength ratings[1]. We also assume the attacker has access to the checker and obtained another dataset of plain text passwords leaked from other websites as prior knowledge, which is used to train the password crackers. Since the attacker does not know the correlation between the plain text and the hashed passwords, a straightforward strategy is to assume a common

[1]In general, strength checkers can accept passwords of any ratings from "weak" to "strong", but there are only several ratings available.

distribution in both datasets and use all the plain text passwords to train the cracking model. However, the target passwords might have been created mostly by users who trust the strength feedback from the checker and create passwords only if they are labelled as "strong". Then, the target passwords are reasonably dissimilar from the training passwords which come from other sources. Therefore, to compromise such biased target passwords, the attacker will likely have better cracking results if the training passwords are also "strong".

In our experiment, the objective is to see if more "strong" passwords in the target dataset can be compromised when the attacker uses the password strength checker to select training data. Figure 1 summarizes the evaluation process. First, we randomly select 50% of the passwords from a dataset in Table 1 to be the *Nonselective Training* dataset. Then, we apply a password strength checker in Table 2 to score each password in the *Nonselective Training* dataset, and select only those passwords labelled as "strong" to make up the the *Selective Training* dataset. From the other 50% of the passwords, we apply the same checker selection method to build the *Testing* dataset. Finally, we use *Nonselective Training* and *Selective Training* datasets separately, as input to JtR, OMEN, and PCFG, to crack the *Testing* dataset.

In Table 2, we show the percentages of selected passwords from the datasets, e.g., *Bloomberg-Train* and *Bloomberg-Test* indicate the percentages of "strong" passwords marked by Bloomberg's checker in the datasets from which we sample training and testing data, respectively. Since we randomly divide an original dataset into halves, the distributions of "strong" passwords in both halves are approximately the same.

To conduct a comprehensive and comparable evaluation, we perform passwords cracking in both *Intra-site* and *Cross-site* scenarios. In *Intra-site* cracking, the training data and target data come from the same original dataset and in *Cross-site* cracking, the training data is from a different dataset. To make the comparison fair, we limit each cracking session to 10 billion passwords guesses uniformly. We present

intra-site cracking results of Renren, LinkedIn, Rockyou, and Tianya in Figure 2, and *cross-site* cracking results with Bloomberg's password checker in Table 3.

In Figure 2, the *intra-site* results show that *Selective Training* enable all the cracking algorithms to compromise much more "strong" passwords than *Nonselective Training*. Figure 2 (a) shows the cracking scenario where the passwords are selected by Bloomberg's checker. The performance gain of using *Selective Training* is significant. Specifically, regarding PCFG, with *Nonselective Training*, it can only crack 0.07%, 4.58%, 0.22%, and 0.01% of the passwords in the target data from Renren, LinkedIn, Rockyou, and Tianya, respectively, whereas with *Selective Training*, it can crack 31.15%, 15.40%, 24.78%, and 14.37%, respectively.

Figure 2 (b) shows the cracking scenario where QQ's checker is used. Although *Selective Training* can boost the cracking capability uniformly, the performance gain is smaller compared to that with Bloomberg's checker. For Tianya, we see that the cracking results of *Nonselective Training* and *Selective Training* are almost the same when JtR and PCFG are in use. The likely reason for this phenomenon is that QQ's checker is not as stringent as Bloomberg's, thus having less bias on the selected "strong" passwords. Another interesting observation is that PCFG, in Figure 2 (a) and (b) has very different performance. It shows much more performance gain when Bloomberg's checker is used. Due to PCFG's nature, this confirms that Bloomberg's checker is more stringent on password structure than QQ's checker.

In Table 3, we show the results of *cross-site* cracking with Bloomberg's password checker. Surprisingly, we see that the cracking performance with *Selective Training* is uniformly and significantly better without exception. Gamigo, as a typical dataset with German linguistic patterns, is also subjective to a greater cracking enhancement when the adversary uses the checkers to select training data from a Chinese or English dataset e.g., a performance gain of up to 24% is observed when training from Rockyou

and cracking with PCFG. This means that regardless of where attackers obtain passwords for training, they can always improve their cracking capability drastically by using the password checker associated with the target data to make a good selection of training data [2].

Our attack-based evaluation is meaningful in the following ways. We do not make assumptions on what datasets the attacker possesses. We show that as long as the corresponding password checker of the target dataset exists, the attacker can successfully crack more passwords in the target dataset that are labelled as "strong". In our experiment, *Nonselective Training* represents the original dataset that the attacker has, without applying any selection. This makes sense as the attacker will not have prior knowledge of how to select the training data simply because the target dataset is hashed. When the password checker is available, it provides information for the attacker about the target dataset, thus enabling them to select training data accordingly. Therefore, it is meaningful to compare the cracking performance with and without the password checker.

The testing dataset represents the target dataset that attackers aim to compromise, which in our case is limited to only passwords rated "strong" by the password checkers. This can be applied to passwords of any ratings, e.g., "moderate", "weak". Although we do not have Bloomberg or QQ's password datasets, by using their checkers to sample data from the available datasets, we can regard the selected data as their fair representatives.

To better understand how the adversary can leverage commercial password strength checkers, we conduct attack-based evaluation in Figure 6 on more password strength checkers in PARS [55]. In Figure 4, 5, 6, 7, and 8, we show the results of evaluating QQ, Target, Twitter, Yahoo! and 12306.cn's password strength checkers, respectively,

[2]Of course, as previous work has shown, choosing a training dataset that has similar characteristics as the target dataset is also important to optimize cracking (e.g., choose a Chinese dataset for training if the target dataset is likely Chinese).

Table 4: Password Cracking (QQ's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	0.22%	1.85%	0.70%	3.78%	7.24%	6.73%	3.10%	5.64%	5.87%	9.29%	7.64%	7.57%	0.58%	4.64%	2.43%	6.59%	7.51%	7.51%
LinkedIn	0.20%	2.12%	0.61%	4.11%	6.88%	8.22%	3.77%	7.96%	7.69%	14.74%	9.36%	11.20%	0.63%	5.98%	2.45%	7.90%	7.33%	10.79%
Rockyou	4.15%	37.80%	5.28%	13.48%	28.32%	31.57%	3.29%	5.76%	6.19%	9.58%	7.81%	9.56%	1.19%	6.85%	3.30%	7.99%	6.55%	10.18%
Tianya	0.45%	3.53%	1.49%	7.81%	11.17%	15.50%	5.63%	9.76%	11.31%	17.87%	16.17%	19.97%	2.01%	10.78%	7.10%	17.50%	11.90%	21.20%
Gamigo	0.37%	0.85%	0.43%	1.44%	0.79%	1.58%	0.46%	0.80%	0.99%	1.58%	1.08%	1.86%	0.25%	0.87%	0.51%	1.33%	0.69%	2.04%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	0.05%	0.16%	0.07%	0.67%	6.51%	5.65%	0.74%	1.84%	2.08%	9.16%	7.30%	7.15%
LinkedIn	0.06%	0.28%	0.06%	0.72%	6.32%	6.40%	0.63%	1.61%	1.26%	8.86%	6.95%	6.76%
Rockyou	0.91%	2.57%	0.89%	6.95%	5.75%	7.48%	0.73%	1.68%	1.55%	7.36%	6.21%	6.05%
Tianya	0.14%	0.44%	0.16%	1.68%	9.89%	10.90%	1.07%	2.36%	2.28%	12.58%	11.27%	10.98%
Gamigo	50.07%	50.21%	1.59%	20.69%	0.84%	1.85%	0.11%	0.32%	0.25%	1.08%	0.59%	0.53%

Table 5: Password Cracking (Target's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	0.00%	0.09%	0.14%	3.33%	11.39%	10.00%	0.00%	0.23%	1.32%	8.16%	11.95%	13.89%	0.00%	0.08%	0.38%	8.97%	12.65%	14.11%
LinkedIn	0.00%	0.02%	0.03%	1.11%	3.89%	3.78%	0.00%	0.36%	0.95%	6.11%	4.29%	7.45%	0.00%	0.02%	0.09%	3.49%	4.48%	6.52%
Renren	0.00%	0.22%	0.49%	6.69%	8.74%	8.85%	0.00%	0.11%	1.24%	7.45%	9.23%	10.79%	0.00%	0.00%	0.59%	8.15%	9.55%	11.17%
Rockyou	0.00%	0.06%	0.16%	2.43%	6.51%	5.81%	0.00%	0.14%	0.90%	5.52%	7.00%	8.47%	0.00%	0.06%	0.43%	8.07%	7.43%	8.47%
Tianya	0.01%	0.13%	0.25%	1.64%	1.55%	1.08%	0.00%	0.06%	0.35%	1.83%	1.02%	1.41%	0.00%	0.02%	0.22%	2.18%	1.13%	1.41%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	0.00%	0.01%	0.04%	1.57%	7.64%	12.88%	0.00%	1.26%	0.46%	10.84%	12.13%	14.46%
LinkedIn	0.00%	0.01%	0.00%	0.55%	2.66%	4.52%	0.00%	0.22%	0.04%	2.41%	4.15%	5.18%
Renren	0.05%	0.70%	0.32%	6.69%	5.83%	10.79%	0.00%	0.65%	0.11%	5.83%	9.01%	10.36%
Rockyou	0.00%	0.01%	0.02%	1.22%	4.22%	7.11%	0.00%	0.42%	0.12%	4.31%	6.82%	7.74%
Tianya	0.01%	0.44%	0.18%	7.23%	1.83%	3.13%	0.01%	0.23%	0.15%	1.47%	0.96%	1.30%

which are consistent with out observation and conclusions.

Remarks. In this chapter, we conduct a comprehensive evaluation to study the feasibility and effectiveness for attackers to use existing commercial password strength checkers to launch more powerful attacks. The results are surprising that commercial password checkers can actually significantly help attackers compromise more "strong" passwords rated by the password strength checkers. This means that if the users trust a password strength checker and always creates "strong" passwords, their accounts are

Table 6: Password Cracking (Twitter's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	0.14%	0.91%	0.16%	1.71%	1.07%	11.45%	0.90%	3.54%	2.16%	6.97%	3.68%	14.39%	0.47%	3.15%	1.09%	5.65%	1.05%	14.81%
LinkedIn	0.12%	1.24%	0.05%	0.97%	0.03%	9.12%	1.58%	6.85%	3.03%	11.97%	4.51%	14.44%	0.30%	3.39%	0.37%	3.50%	0.04%	11.93%
Renren	8.15%	17.99%	12.32%	24.15%	0.07%	19.19%	1.03%	2.77%	1.57%	4.33%	1.65%	7.30%	0.52%	7.47%	0.78%	10.54%	0.20%	6.97%
Rockyou	0.21%	1.56%	0.17%	2.12%	0.04%	10.35%	2.01%	5.28%	2.98%	8.46%	4.83%	13.43%	1.09%	6.14%	2.10%	8.82%	0.12%	13.70%
Tianya	0.59%	2.00%	0.53%	3.15%	0.01%	6.27%	1.10%	2.53%	1.78%	3.98%	1.25%	7.84%	0.47%	2.52%	0.72%	3.02%	0.11%	7.05%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	0.04%	1.07%	0.00%	0.51%	0.82%	8.49%	0.02%	4.54%	0.29%	2.34%	0.04%	14.91%
LinkedIn	0.09%	1.83%	0.04%	0.37%	0.02%	5.82%	0.00%	3.96%	0.02%	0.27%	0.11%	10.87%
Renren	0.75%	8.57%	0.62%	8.68%	0.05%	6.01%	0.00%	2.08%	0.03%	0.25%	0.07%	5.55%
Rockyou	0.20%	1.95%	0.08%	0.85%	0.03%	5.30%	0.01%	3.49%	0.09%	0.32%	0.19%	11.80%
Tianya	1.00%	5.22%	0.62%	5.29%	0.02%	9.33%	0.01%	1.91%	0.05%	0.22%	0.04%	5.46%

Table 7: Password Cracking (Yahoo's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	1.23%	6.47%	1.09%	8.03%	8.55%	8.07%	5.97%	8.83%	7.28%	10.80%	8.89%	8.68%	2.04%	8.39%	3.49%	9.46%	8.85%	8.71%
LinkedIn	1.12%	7.81%	0.91%	8.89%	7.86%	9.77%	7.38%	12.39%	8.85%	15.74%	9.95%	11.44%	2.12%	10.98%	3.32%	11.27%	8.35%	11.35%
Renren	8.19%	22.32%	9.03%	27.74%	14.69%	22.22%	13.93%	18.97%	14.08%	19.25%	15.99%	16.50%	6.64%	20.82%	8.12%	18.54%	15.08%	18.51%
Rockyou	4.69%	18.72%	3.42%	19.75%	16.73%	22.66%	18.58%	24.19%	18.17%	25.16%	20.54%	24.18%	10.01%	26.32%	13.15%	29.37%	17.76%	27.06%
Tianya	1.24%	3.59%	1.10%	4.14%	2.14%	3.24%	2.44%	3.50%	2.69%	3.99%	2.49%	3.60%	1.10%	3.53%	1.42%	3.77%	1.97%	4.16%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	0.41%	1.98%	0.11%	0.89%	7.51%	6.04%	3.02%	5.22%	2.55%	10.48%	8.57%	8.38%
LinkedIn	0.41%	2.43%	0.09%	0.90%	6.89%	6.52%	3.00%	5.41%	1.53%	9.54%	7.87%	7.61%
Renren	2.87%	10.19%	1.26%	7.37%	12.01%	11.31%	7.32%	11.40%	3.55%	14.48%	14.29%	13.19%
Rockyou	1.93%	7.61%	0.40%	2.44%	13.03%	11.25%	8.76%	13.21%	3.98%	16.77%	16.71%	16.24%
Tianya	1.42%	3.33%	3.67%	46.36%	2.21%	4.25%	1.00%	2.06%	0.72%	2.74%	1.67%	1.43%

Table 8: Password Cracking (12306's Password Checker).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	1.78%	24.57%	1.17%	6.76%	0.00%	32.19%	6.02%	33.66%	7.43%	32.37%	16.40%	38.33%	6.39%	33.54%	7.43%	31.08%	0.12%	40.54%
LinkedIn	0.41%	12.59%	0.28%	2.74%	0.00%	16.44%	2.17%	22.72%	4.30%	24.84%	8.06%	23.59%	1.98%	17.60%	2.08%	13.19%	0.01%	22.51%
Renren	4.25%	30.10%	4.62%	15.09%	0.00%	26.91%	7.58%	27.60%	7.66%	24.72%	9.55%	31.99%	8.26%	29.57%	9.10%	24.41%	0.15%	33.51%
Rockyou	2.23%	24.20%	1.26%	8.57%	0.00%	25.94%	7.28%	27.71%	7.40%	25.60%	11.48%	30.01%	9.17%	31.42%	10.68%	30.78%	0.02%	31.52%
Tianya	1.61%	17.38%	1.78%	6.65%	0.00%	13.36%	2.23%	15.96%	3.16%	12.53%	4.79%	18.51%	2.58%	15.91%	2.90%	11.06%	0.03%	21.57%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Gamigo	1.29%	22.17%	0.43%	8.72%	0.00%	36.00%	1.35%	35.20%	1.11%	19.66%	0.06%	35.57%
LinkedIn	0.31%	12.13%	0.11%	3.48%	0.00%	19.34%	0.16%	19.01%	0.24%	6.55%	0.03%	18.00%
Renren	5.46%	31.99%	4.55%	21.23%	0.00%	33.43%	0.91%	26.84%	0.91%	13.04%	0.08%	24.34%
Rockyou	1.73%	21.75%	0.00%	9.02%	0.00%	28.73%	1.25%	28.01%	0.68%	14.65%	0.09%	27.02%
Tianya	3.60%	25.29%	0.00%	22.90%	0.00%	26.69%	0.23%	14.30%	0.40%	4.26%	0.02%	11.41%

still more vulnerable to training-based attacks when the adversary obtains the checker. Since the training-based crackers only become more powerful when the training data is more similar to the target data, in our evaluation model, the *Selective Training* data is more similar to the *Testing* data, which further implies that the checkers exert bias on the selected passwords. Due to the nature that password policies and scoring mechanisms are static, the password distribution is consistently biased. Such bias, while not necessarily enforcing good password strength, poses significant threats on the overall password dataset security. Therefore, it is meaningful to address this limitation of the password strength checkers, and investigate how to enhance overall password data security.

CHAPTER IV

DYNAMIC PASSWORD POLICY GENERATOR

One could argue that a potential solution to the password checker limitations is to have better web technologies to hide the policies and detect malignant password strength querying. However, it can result in delay in strength feedback and high false-positive rates in detection. Further, it does not resolve the fundamental bias in password distribution. Therefore, we take another approach to the problem and explore the feasibility of providing dynamic password policies to users. Considering usability, rather than forcing all users to create extremely complex passwords, we focus on the overall strength of the password dataset and ensure that the passwords created by the users have diversity (i.e., cover the vast majority of the entire password space uniformly). In this chapter, we propose the Dynamic Password Policy Generator, namely *DPPG*, as an alternative to traditional password strength checkers.

4.1 Overview

DPPG is a diversity-based and database-aware application that generates password creation policies dynamically for the users. Instead of purely focusing on the complexity of candidate passwords, *DPPG* enforces a baseline complexity on the passwords (e.g., more than 6 characters long) to protect them from simple attacks, e.g.,

Table 9: Password Policy Requirement Types.

Type	Description
Length	use a range of password length
Composition	use a number of different character types
Alternation	use a number of character type transitions
Good Chars	include specific characters
Bad Chars	exclude specific characters
Structure	use a specific structure

dictionary, brute-forcing. However, more focus is put on protecting the password distribution within a database by preventing aggregation of similar passwords that form a characteristically biased distribution. As long as a candidate password meets the policy, it is accepted and no additional strength feedback is provided. The policies are generated to search for candidate passwords that balance the password characteristics distribution. The underlying diversity-based metric implemented in *DPPG* is further elaborated in Chapter 5.

In Figure 3, we show how *DPPG* works. Initially, system administrators can place complex or random passwords as seeds in the password database. The seeds can form a white list to inject certain desired password characteristics, e.g., structures, n-grams. Based on the seeds, *DPPG* can start to generate password policies to users. Since dynamically generating policies requires necessary computational time depending on the number of existing passwords, to avoid delay in responding to users' requests, a policy queue is used to store policies as a buffer each time when a batch of policies are created. When the size of the policy queue reduces below a threshold, e.g., 25%, *DPPG* is signalled to generate new policies.

4.2 Two Modes: Explore and Exploit

In order to intelligently generate password policies based on the current password distribution, *DPPG* maintains a global characteristics frequency map and a history of generated password policies[1] that can approximate the current password distribution. There are two modes for *DPPG* to expand the usable password space and balance the current password distribution.

The `exploration` mode mainly aims to expand the password space by actively introducing new characteristics based on the global characteristics frequency map. Before an incoming password is hashed, *DPPG* extracts its characteristics and stores

[1]No plain text passwords are stored.

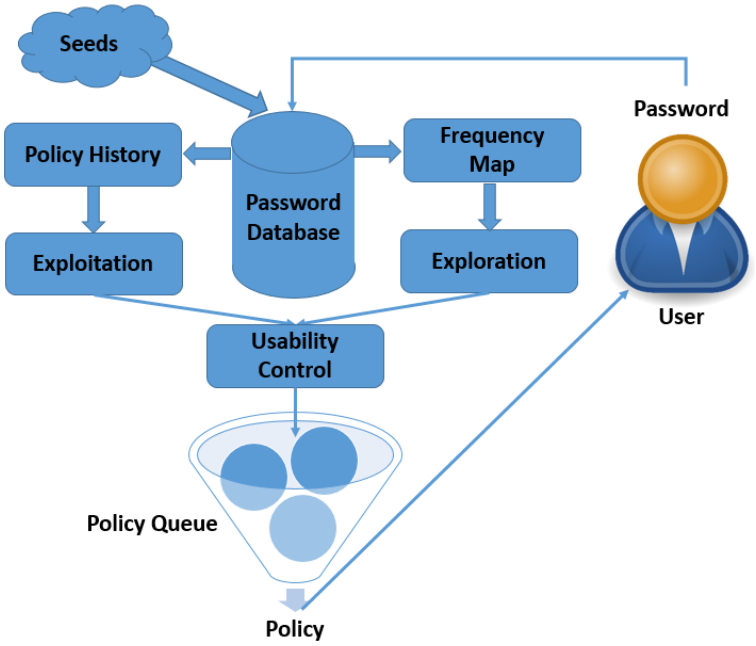


Figure 3: Dynamic Password Policy Generator

the metadata in the frequency map, which keeps tracks of the overall distribution of password attributes e.g., frequency of *structures*, *characters*, and denotes the current password space. In `exploration` mode, *DPPG* creates policies that require users to be more "creative" in making a password e,g., using the character "(" which is not usual even in special characters. In this way, the passwords can cover a larger textual search space than the regular human linguistic patterns. Initially when there are not many passwords, a random mechanism is adjusted to launch the `exploration` mode more often to aggressively enlarge the password space. When the password characteristics distribution is relatively uniform as observed from the `exploitation` mode, the `exploration` mode is also evoked to introduce new characteristics.

Since purely expanding the password space is equivalent to making random passwords, *DPPG* also relies on another major component. The `exploitation` mode aims to enhance password diversity and balance the current password distribution, with the help of the password policy history. Since passwords are hashed in the

database, *DPPG* stores previously generated password policies to approximate current password distributions and analyze the password diversity through the metric and algorithm discussed in Chapter 5. *DPPG* then identifies password characteristics that exist in the database with low appearance frequencies, and generate policies that require such characteristics. Therefore, *DPPG* creates policies that are usable and balance the password distribution by temporarily increasing the frequencies of less common password attributes.

Based on the two modes, *DPPG* determines the critical characteristics requirements, but only renders the final policies after passing them to the usability control module. In our implementation, there are 6 types of requirements that can form a policy as shown in Table 9, and the usability control module is evoked to ensure that the final policies contain only a reasonable number of requirements and are in different formats as shown in examples below.

Include the character(s): 'v', 'Z'

Avoid the character(s): a, s, e

Use the structure: LLLLLUUS

Number of characters: 8 to 12 (inclusively)

Number of character types: 4

Number of alternations: 3 to 4 (inclusively)

Include the character(s): '?', 'U', ')'

4.3 Usability Analysis

The usability of *DPPG* can translate into the ability for users to follow the policies and maintain the passwords they create. In this chapter, we test *DPPG* on real users and collect passwords for further analysis.

Recruitment. After our protocol was approved by the Institutional Review Board (IRB), we conducted a usability test of *DPPG* on Amazon Mechanical Turk [1].

Table 10: Mechanical Turk User Study.

DPPG											
Times	0	1			2			3			
Session	0	2	3	4	2	3	4	2	3	4	
#	74	3	0	1	1	1	2	1	1	1	
%	91.36%	4.94%			2.47%			1.23%			
QQ Checker											
Times	0	1			2			3			
Session	0	2	3	4	2	3	4	2	3	4	
#	75	2	0	1	2	1	3	0	0	0	
%	92.59%	3.70%			3.70%			0.00%			

We restricted the participants to a qualification type that requires at least 95% of approval rate, and 500 approved tasks. We excluded minors and only included English speakers. Our recruitment statement on Mechanical Turk is attached in the appendix.

Protocol. Our approach is to test if users who create their passwords by *DPPG* policies can successfully remember them for a reasonable time period. We also require the same participants to create passwords using QQ's password strength checker as the control group. The participants are not informed of the purpose of our study or anything introduced in this thesis. Each participant who accepts our human intelligence task (HIT) on Amazon Mechanical Turk is asked to access our web server with registration and login services. Participants are asked to complete 4 sessions of experiments which are separated by time intervals of 24 hours, 48 hours, and 72 hours to finish the entire study.

In the first session, the participant is directed to visit two artificial websites to register two accounts with usernames and passwords, following a *DPPG* policy and using QQ's checker, respectively. Then the participant simply concludes the session by logging into the accounts with the credentials they just created. For the rest of the sessions, participants simply return to our web interface during the time specified at the end of each previous session and logged into the accounts with their credentials. All participants are informed in the beginning of the study that forgetting their passwords during the study was fine and would not penalize them. If they did forget their passwords, they were prompted to make new ones.

Due to our task requirements, participants are involved for 6-7 days to attend all sessions. The total in-session time is around 12 minutes. We paid $1.5 to each participant who completed all sessions in time. In order to collect more passwords for further analysis, we made the tasks on mechanical turk viewable to all qualified users who can attempt using *DPPG* before deciding to join the study. We also hosted standalone sessions purely to collect passwords from users. Although passwords are stored in plain text for future analysis, they are not visible to *DPPG* which only approximates password distribution by the history of policies. We present the details of the user-study documents e.g., consent form, session screen shots in the appendix.

Results. After we conducted our study on Amazon Mechanical Turk for 1.5 months, there are 115 users who accepted our study and 81 of them finished 4 sessions completely. Since we do not keep track of participants' email addresses for privacy reasons, we do not explicitly survey those who dropped out of the study. We show the results based on the records of these 81 participants in Table 10, where *Times* denotes the number of sessions where participants failed to log in with the correct passwords after some trials, and *Session* denotes the indices of the sessions where users re-created passwords. The column with 0 in *times* and *session* indicates participants who logged in all sessions successfully. From the results, we see that 74 participants out of 81 consecutively succeeded in logging into our sessions with the right passwords they created according to the policies thus demonstrating the ability to remember the passwords up to a week. Of the 7 participants who forgot their passwords in at least one session, 4, 2, and 1 of them had to re-create their passwords in exactly 1, 2, and 3 sessions, respectively. Most of the participants who had to re-create passwords in one session did it in session 2 and if the participant successfully logs in in session 2, it is almost certain for them to pass the rest of the sessions. QQ's checker as the control group shows very similar statistics but is slightly better. This demonstrates that policies generated dynamically from *DPPG* are usable, in terms of the ability of

users to maintain the passwords.

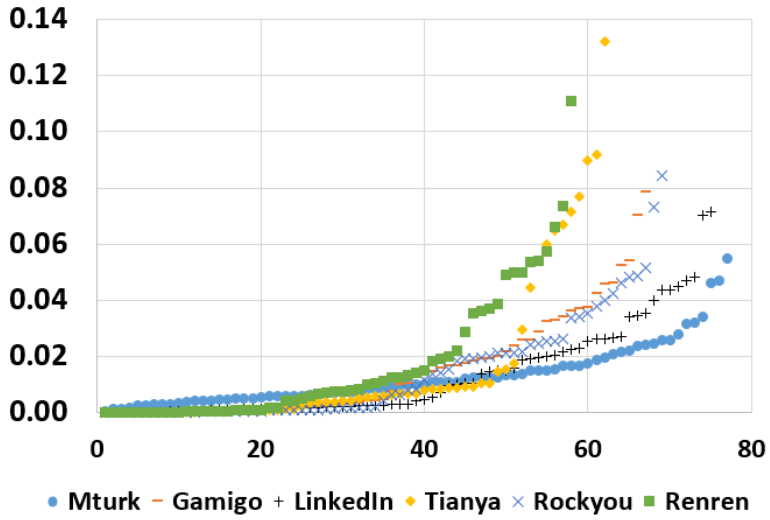
Furthermore, the passwords from *DPPG* and QQ's checker share 54 out of 90 passwords in common. Since password policies from *DPPG* are dynamic, unpredictable and in various templates, a more likely explanation for this phenomenon is that about half of the users reused passwords created with *DPPG* for the QQ' checker. Although reusing passwords is a bad practice, this implies that users either reuse the passwords to get strong strength feedback in QQ's checker, or to better maintain the passwords.

In a final survey, we further obtain subjective feedback on the usability of *DPPG* and QQ's checker. When asked about the ease in following the policies or strength feedback, 63.75% and 73.75% of participants thought *DPPG* and QQ's checker, respectively were above average. In addition, 65.43% and 61.73% thought *DPPG* and QQ's checker, respectively enable them to make more secure passwords. Finally, 77.78% of participants indicate the passwords they created with *DPPG* are drastically different from their other passwords and the rest indicated the passwords are somewhat similar but not the same. This shows that *DPPG* helps reduce the frequency of reusing passwords because users are unlikely to leverage their other passwords to satisfy the dynamic policies.

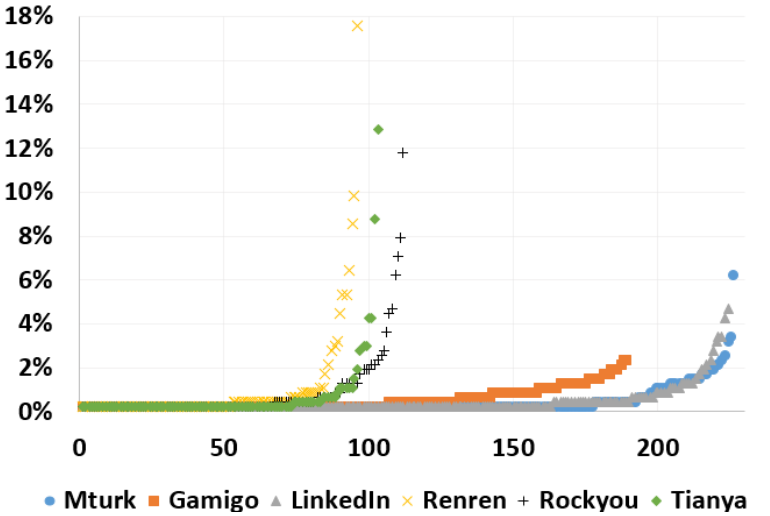
4.4 Passwords Evaluation

Using a total of 467 passwords collected from our usability study and other standalone sessions, which we denote as the Mturk dataset, we provide a statistical analysis of the password characteristics. In order to have a comparison with passwords created without *DPPG*, we randomly sampled the same number of passwords from datasets shown in Table 1 to form other testing datasets.

To compare the password space in the testing datasets, we conduct a character distribution analysis shown in Figure 4 (a). We assign each unique character existing in a testing dataset with a number ordered by its appearance frequency. From the



(a) Character Distribution



(b) Structure Distribution

Figure 4: Mturk Password Analysis.

plot, we see that the Mturk dataset contains more unique characters in its passwords than any of the other testing datasets. The character distribution of the Mturk dataset is also more uniform than the other datasets, suggesting that *DPPG* can expand the password space while also enforcing a balanced password distribution.

In Figure 4 (b), we conduct a similar analysis on the password structure. Again, the Mturk dataset demonstrates the largest variety of password structures among all the testing datasets and a fairly balanced distribution. Such diversity in password

structure is meaningful and implies that a structure-based cracking algorithm like PCFG will be less efficient in cracking the datasets, because all structures are almost equally likely.

To further compare the security of the Mturk dataset and other testing datasets, we employ an attack-based analysis using the datasets in Table 1 excluding the data in the testing datasets for training to crack the Mturk and other testing datasets with 10 billion guesses. To eliminate the bias due to a small sample size of the testing data, we re-sample testing datasets and crack them in 10 repeated sessions to obtain the average cracking rates. Table 11 shows the partial cracking results of Mturk, LinkedIn, and Renren datasets, and the full results are in the appendix. As we see, compared to other testing datasets, the Mturk dataset is much less vulnerable to the attacks. This is consistent with the results in Figure 4. It also shows that passwords created with *DPPG* policies are more diverse and dissimilar to other passwords, thus being more secure from training-based cracking.

One interesting observation is that although the LinkedIn dataset has a very close structure distribution with Mturk dataset in Figure 4 (b), and also a sub-optimal character distribution in Figure 4 (a), it is still noticeably more vulnerable to cracking than the Mturk dataset. Since the performance of the training-based cracking algorithms mainly depends on the training data as shown in previous work, it means that our training datasets are much more similar to the LinkedIn target sample than to the Mturk dataset.

In Table 12, we show more results on cracking analysis on the Mturk dataset and the samples from datasets shown in Table 1. We see that cracking performance on the Mturk dataset is much worse than the other datasets, which is consistent with results shown in Figure 4 and conclusions in this section.

Table 11: Cracking Evaluation on Mturk Passwords

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	4.93%	7.92%	7.49%	7.49%	7.28%
LinkedIn Sample	22.27%	26.34%	19.49%	23.98%	11.35%
Renren Sample	59.31%	63.60%	71.52%	67.02%	68.95%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	5.35%	7.49%	7.49%	7.71%	6.00%
LinkedIn Sample	11.35%	24.84%	11.13%	21.20%	3.64%
Renren Sample	41.97%	56.75%	62.96%	60.17%	53.10%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	2.57%	3.43%	3.00%	3.43%	4.71%
LinkedIn Sample	17.77%	19.91%	17.99%	19.06%	17.13%
Renren Sample	28.91%	30.62%	50.54%	44.11%	52.46%

Table 12: Mturk Dataset Evaluation.

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	4.93%	7.92%	7.49%	7.49%	7.28%
Gamigo Sample	16.06%	17.34%	14.56%	17.34%	9.85%
LinkedIn Sample	22.27%	26.34%	19.49%	23.98%	11.35%
Renren Sample	59.31%	63.60%	71.52%	67.02%	68.95%
Rockyou Sample	57.82%	62.10%	54.39%	64.03%	46.68%
Tianya Sample	57.17%	62.53%	71.73%	65.31%	73.88%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	5.35%	7.49%	7.49%	7.71%	6.00%
Gamigo Sample	13.06%	17.13%	9.42%	16.92%	4.28%
LinkedIn Sample	11.35%	24.84%	11.13%	21.20%	3.64%
Renren Sample	41.97%	56.75%	62.96%	60.17%	53.10%
Rockyou Sample	42.18%	53.53%	41.97%	62.74%	20.34%
Tianya Sample	45.40%	57.39%	65.10%	62.53%	62.31%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	2.57%	3.43%	3.00%	3.43%	4.71%
Gamigo Sample	10.49%	10.92%	10.71%	12.85%	11.13%
LinkedIn Sample	17.77%	19.91%	17.99%	19.06%	17.13%
Renren Sample	28.91%	30.62%	50.54%	44.11%	52.46%
Rockyou Sample	40.90%	38.54%	43.25%	47.97%	41.11%
Tianya Sample	25.48%	26.98%	43.04%	42.40%	59.96%

4.5 Prevention and Detection of Misuse

While *DPPG* aims to protect users' passwords, it is important to understand how it can be misused and leveraged by the adversary. It would be no surprise if the adversary can eventually hack the server of *DPPG* and obtain crucial information on the password distribution. In this section, we discuss the security of *DPPG* and the impact of possible misuse.

First we assume the adversary seeks to obtain crucial information on the password distribution by monitoring the dynamic password policies. Since the policies are designed to request diverse passwords that are dissimilar to the existing passwords, they only reflect password characteristics that are rare or absent, which is not useful information for cracking. In fact, the general password space is approximately infinite, and the usable passwords created by people only take a smart part of it. Therefore, the adversary cannot infer the current password distribution from the policies. The adversary may try to record the password policies they see by not submitting a password during registration so as to learn what policies other users might follow. For unfulfilled password policies, *DPPG* is designed to expire such policies within a short period of time, by artificially creating dummy passwords according to the policies. Such dummy passwords can be selected from leaked password data.

Next, we consider the scenario where the adversary can hack into the hosting server and obtain subsets of password hashes and the metadata used in *DPPG*. The most sensitive information from the generator is the global frequency map, which stores the appearance frequencies of password characteristics. While the dynamic policies presented to users reveal password characteristics not in the database, the metadata that is leaked may expose the characteristics in the database. However, the global frequency map will ideally present a nearly uniform distribution because *DPPG* tries to constantly balance the current distribution. The adversary is likely to see password structures with similar possibilities as shown in Figure 4, and has

to enumerate all possible cases. Furthermore, no semantic or positional information is stored in the map. Therefore, the adversary can only leverage this information minimally and will still need to put a significant amount of effort in cracking the password hashes. This scenario is further discussed in Chapter 5.4 when we evaluate the cracking performance on diverse passwords.

Remarks. To the best of our knowledge, *DPPG* is the first password policy generator that can generate password policies dynamically according to the current password distribution. Since no password strength feedback is returned and the policies generated by *DPPG* are dynamic and unpredictable, the attacker will find it extremely difficult, if not impossible, to learn the system or the inner password distribution. The policies themselves are in different formats and only contain information that is ideally contrary to the distribution in the database, because *DPPG* always tries to balance the current distribution and expanding password space. Through the characteristics analysis and the attack-based evaluation, we further verify that password datasets created with *DPPG* are diverse and relatively robust to training-based cracking attacks. Furthermore, the usability of *DPPG* is not sacrificed for dynamic policies according to our user study, which makes *DPPG* practical to use. Therefore, *DPPG* can be a more secure alternative to current password strength checkers in terms of protecting password distribution information and preventing crafted training-based offline attack.

CHAPTER V

PASSWORD DIVERSITY

In this chapter, we propose to measure the strength of a password dataset in terms of password distribution, by evaluating the password diversity in the dataset.

We define password diversity as within a password dataset, how dissimilar passwords are with each other regarding a specific set of characteristics. For example, "*forgetme886*" and "*iloveyou775*" are very similar even though they don't share many common characters. They are similar because they both have 11 characters; they contain only lower-case English alphabets and numerical digits; and they are composed by 8 letters followed by 3 digits. If password length, types of characters and structure are the characteristics of individual passwords used to determine similarity, we can claim these two passwords are very similar. However, it is also interesting to point out that, if we want to consider more sophisticated characteristic such as semantics, the actual meaning of words in the passwords can conversely make them less similar. Therefore, the similarity should be a conglomerate measure of all password properties of interest, rather than a measure of a single or typical attribute.

In a password dataset, the distribution of such characteristics can then be used to describe the diversity of the passwords. If the distribution is closer to a uniform

Table 13: Password Attributes.

Attribute	Type	Weight	Function
`Length`	absolute	w_1	$f^1_{p_{ij}}$
`Comp`	absolute	w_2	$f^2_{p_{ij}}$
`Alt`	absolute	w_3	$f^3_{p_{ij}}$
`CompFreq`	absolute	w_4,w_5,w_6,w_7	$f^4_{p_{ij}}$,$f^5_{p_{ij}}$,$f^6_{p_{ij}}$,$f^7_{p_{ij}}$
`LCS`	relative	w_8	$f^8_{p_{ij}}$
`LDist`	relative	w_9	$f^9_{p_{ij}}$
`Alt-Str`	absolute	w_{10}	$f^{10}_{p_{ij}}$
`LCS-Str`	relative	w_{11}	$f^{11}_{p_{ij}}$
`LDist-Str`	relative	w_{12}	$f^{12}_{p_{ij}}$

distribution, the passwords are less similar to each other and the password dataset is more diverse. In this chapter, we will quantify password similarity and provide a systematic way to measure the dataset-wise diversity.

5.1 Password Similarity Measure

To quantify password similarity, we first clarify the characteristics that are considered in our measure in Table 13. The *type* of attributes is *absolute* if the attribute is independent and contribute to restraining the password space, or *relative* if it is dependent of both passwords that are in comparison and does not affect the password space. The *weight* of each attribute is its weight in the password similarity quantification. The *function* associated with each attribute, is a normalized measure of the difference between such attributes in two passwords, $\mathbf{p_i}$ and $\mathbf{p_j}$, when quantifying their similarity. The choices of attributes are elaborated as follows.

`Length` is the number of characters in a password. Almost all password policies and strength checkers enforce a minimum length limit due to brute-force attack.

$$f^1_{p_{ij}} = 1 - \frac{|length\,of\,p_i - length\,of\,p_j|}{\max\{length\,of\,p_i, length\,of\,p_j\}} \tag{1}$$

`Comp` is the number of different character types used in the password. L, U, D, and S represent lower-case characters, upper-case characters, numerical digits, and special characters, respectively. In password policies and checkers, `Comp` is also a popular measure. In previous analysis [113], it is shown that requiring more character types reduces usability of the passwords.

$$f^2_{p_{ij}} = 1 - \frac{|comp\,of\,p_i - comp\,of\,p_j|}{4} \tag{2}$$

`Alt` is short for alternation, which means the number of character switches in a password normalized by the password length. For example, "pssS55" has 3 alternations at "p-s", "s-S", and "S-5" and 2 structural alternations at "s-S", and "S-5". It is meaningful to consider alternation in that it relates to both semantic and structural

information about the password. Furthermore, alternation is another strong factor in limiting the usability of a password. *DPPG* limits the alternations in the policies it generates to make them more usable.

$$f_{p_{ij}}^{3} = 1 - |\frac{alt\,of\,p_i}{length\,of\,p_i - 1} - \frac{alt\,of\,p_j}{length\,of\,p_j - 1}| \tag{3}$$

`CompFreq` is the character type appearance frequency.

$$f_{p_{ij}}^{4-7} = 1 - |CompFreq\,in\,p_i - CompFreq\,in\,p_j| \tag{4}$$

`LCS` stands for longest common substring, which is a relative attribute. For a pair of passwords in comparison, we regard the length of the longest common substring as a shared attribute.

$$f_{p_{ij}}^{8} = 1 - \frac{LCS(p_i, p_j)}{\min\{length\,of\,p_i, length\,of\,p_j\}} \tag{5}$$

`LDist` is Levenshtein Distance, which calculates the minimum number of character changes, through insertion, modification, and deletion, that are needed to transform one password to another.

$$f_{p_{ij}}^{9} = 1 - \frac{LevenshteinDistance(p_i, p_j)}{\max\{length\,of\,p_i, length\,of\,p_j\}} \tag{6}$$

We use $\mathbf{S(p_i)}$ to indicate the structure of $\mathbf{p_i}$, and `Alt-Str`, `LCS-Str`, and `LDist-Str` in Table 13 to account for structural information when quantifying the similarity of two passwords.

$$f_{p_{ij}}^{10} = 1 - |\frac{alt\,of\,S(p_i)}{length\,of\,p_i - 1} - \frac{alt\,of\,S(p_j)}{length\,of\,p_j - 1}| \tag{7}$$

$$f_{p_{ij}}^{11} = 1 - \frac{LCS(S(p_i), S(p_j))}{\min\{length\,of\,p_i, length\,of\,p_j\}} \tag{8}$$

$$f_{p_{ij}}^{12} = 1 - \frac{LevenshteinDistance(S(p_i), S(p_j))}{\max\{length\,of\,p_i, length\,of\,p_j\}} \tag{9}$$

We further define the *similarity score* as

$$\mathbf{D}_{p_{ij}} = \sqrt{\sum_{k=1}^{12} (f_{p_{ij}}^{k})^2 \times w_k} \quad , \quad \sum_{k=1}^{12} w_k = 1.$$

To the best of our knowledge, our quantification of the password similarity is the first attempt to provide a comparable measure on how similar two passwords are with regards to various primitive attributes of the passwords. Different from [108] and [24] where only structure and n-gram, respectively is considered, our quantification takes into account a vector of password attributes and has the flexibility to allow weight adjustment for better performance. By assigning different weights to the password characteristics, researchers can put more focus on the evaluation of specific attributes. This is also potentially helpful when new attack models/algorithms emerge based on a composite of the password characteristics. We use $\frac{1}{12}$ for all weights by default to consider all attributes equally for the purpose of this thesis, and we discuss further on weights selection in the next section. More sophisticated password attributes, e.g., semantics, positions of characters can be added to the quantification of password similarity thus making the approach extensible. Based on this quantification, we further propose a metric and a systematic way to measure the diversity of a password dataset.

5.2 Weights Selection

The weights of different password attributes are important factors in the quantification and provide flexibility for system administrators or password researchers. For example, if one wants to evaluate the strength of a password dataset typically against PCFG, the weights of the structure-related attributes e.g., `Alt-Str`, `LCS-Str`, and `LDist-Str` can be adjusted to higher values while still taking into account other less relevant attributes; if they want to evaluate the strength typically against a Markov attack model like OMEN, the weight of the attribute n-gram can be increased accordingly; or if they want to treat each attribute equally, equal weights i.e., $\frac{1}{12}$, can

be assigned to the attributes.

The careful selection of the proper weight values can be studied using sophisticated machine learning techniques and would be itself an interesting and meaningful research topic. For the purpose of this thesis, we do not delve into very complicated models to try to obtain an absolutely optimal set of weights. Instead, we propose a simple and intuitive way that solves the problem to learn a reasonable selection of the weights shown in Figure 5.

We first initialize the weights vector $\mathbf{W}$, with each element valued at 0.5, which means all attributes in the similarity vector $\mathbf{V}_{p_{ij}}$ are equally considered. Then we need some "ground truth" that tells us if two passwords are really similar. Utilizing the state-of-the-art cracking algorithms and leaked password datasets, intuitively, we claim that the passwords cracked by the algorithms are similar to the passwords used to train the algorithms.

From a leaked password dataset, we randomly select a portion of the passwords as training data denoted as set $\mathbb{T}$, and use the remaining passwords as target data. Then we crack the target data with PCFG, OMEN, and JtR using the same training data and limiting the guess number of each algorithm to 10 billion. We aggregate the cracked passwords into a set denoted as $\mathbb{P}$, and the "ground truth" is thus that all training passwords in $\mathbb{T}$ are similar to all cracked passwords in $\mathbb{P}$. Finally we form a bipartite graph between the two sets and each pair of passwords maintain an edge with a computed similarity vector.

In the final step, we traverse each edge in the bipartite graph and evaluate each similarity vector. For each similarity vector, we sort the attribute values and increase the weights of the top 4 attributes by $\frac{1}{2\times|\mathbb{P}|}$, and decrease those of the bottom 4 attributes by $\frac{1}{2\times|\mathbb{P}|}$. Therefore, we obtain an adjusted weight vector with reasonable values.

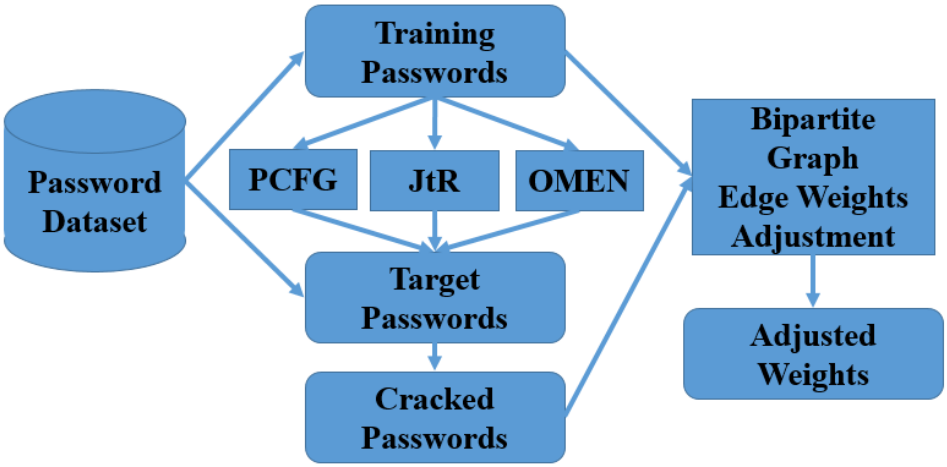


Figure 5: Weights Model

5.3 Diversity-based Metric: Graph Model and Communities

To evaluate the diversity of a sizeable password dataset, we propose to group passwords into communities based on our similarity quantification. A password community contains passwords that have higher similarity with each other, than with passwords in other communities. When password datasets are large, the password diversity can then be represented by the number of communities detected, and the sizes of the communities.

Conceptually and computationally, we connect passwords in a graph model which enables us to analyze the similarities among the passwords. Each password dataset can be built into a graph, with nodes being the passwords, and edges being their relations weighted by the pairwise *similarity score* quantified in the previous section. For a password dataset, we can compute a *similarity score* for each pair of passwords and obtain a weighted complete graph.

The password graph preserves the similarities among the passwords and is convenient for further analysis of password diversity. Since community detection on a complete graph results in overwhelming time and space complexity, we further make the graph sparser by cutting edges that have weights less than a threshold, which is

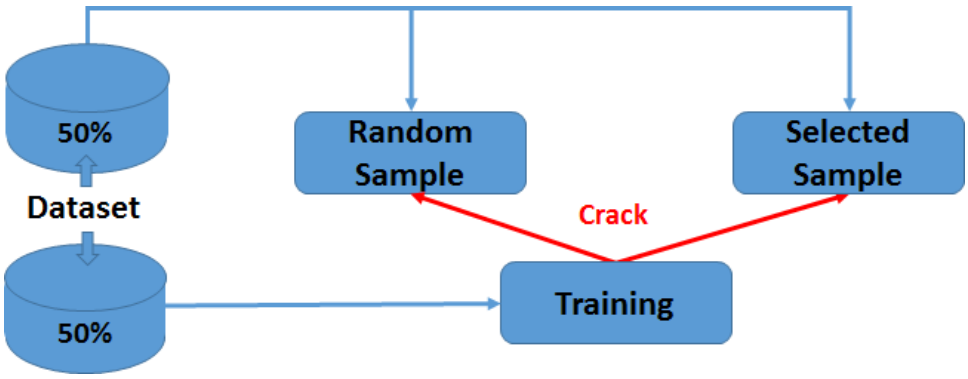


Figure 6: Attack-based Evaluation

Table 14: Diversity-based Password Security Metric.

Dataset	Std/Mean	DivScore	Dataset	Std/Mean	DivScore
Renren	1.63	0.61	Sample	1.97	0.51
Tianya	1.40	0.71	Sample	1.87	0.53
Rockyou	1.27	0.79	Sample	2.03	0.49
LinkedIn	1.11	0.90	Sample	2.23	0.45
Gamigo	0.44	2.27	Sample	2.28	0.44
-	-	-	Mturk	0.38	2.61

by default the mean value of all weights. Then we use a simple, light-weight, and efficient algorithm, the Louvain Method [15], to detect communities in the passwords graph. Finally, we calculate the diversity-based password dataset score **DivScore** by dividing the mean value of the sizes of detected communities by the standard deviation of the sizes. We show that **DivScore** can serve as an indicator of the overall security of a password dataset in the next section. The diversity-based metric also serves as a critical component in the *exploitation mode* of *DPPG*. By analyzing the password policy history and using passwords from leaked datasets, *DPPG* simulates the stored hashed passwords and evaluate the current password diversity to determine the new policy requirements balancing the password distribution.

5.4 Evaluation of the Diversity Measure

Table 15: Cracking Results of the Mturk Dataset.

Training:	Gamigo	LinkedIn	Renren	Rockyou	Tianya
JtR	6.42%	9.85%	7.71%	8.14%	6.85%
OMEN	7.49%	8.57%	7.49%	7.28%	6.00%
PCFG	1.93%	1.71%	1.71%	2.14%	1.93%

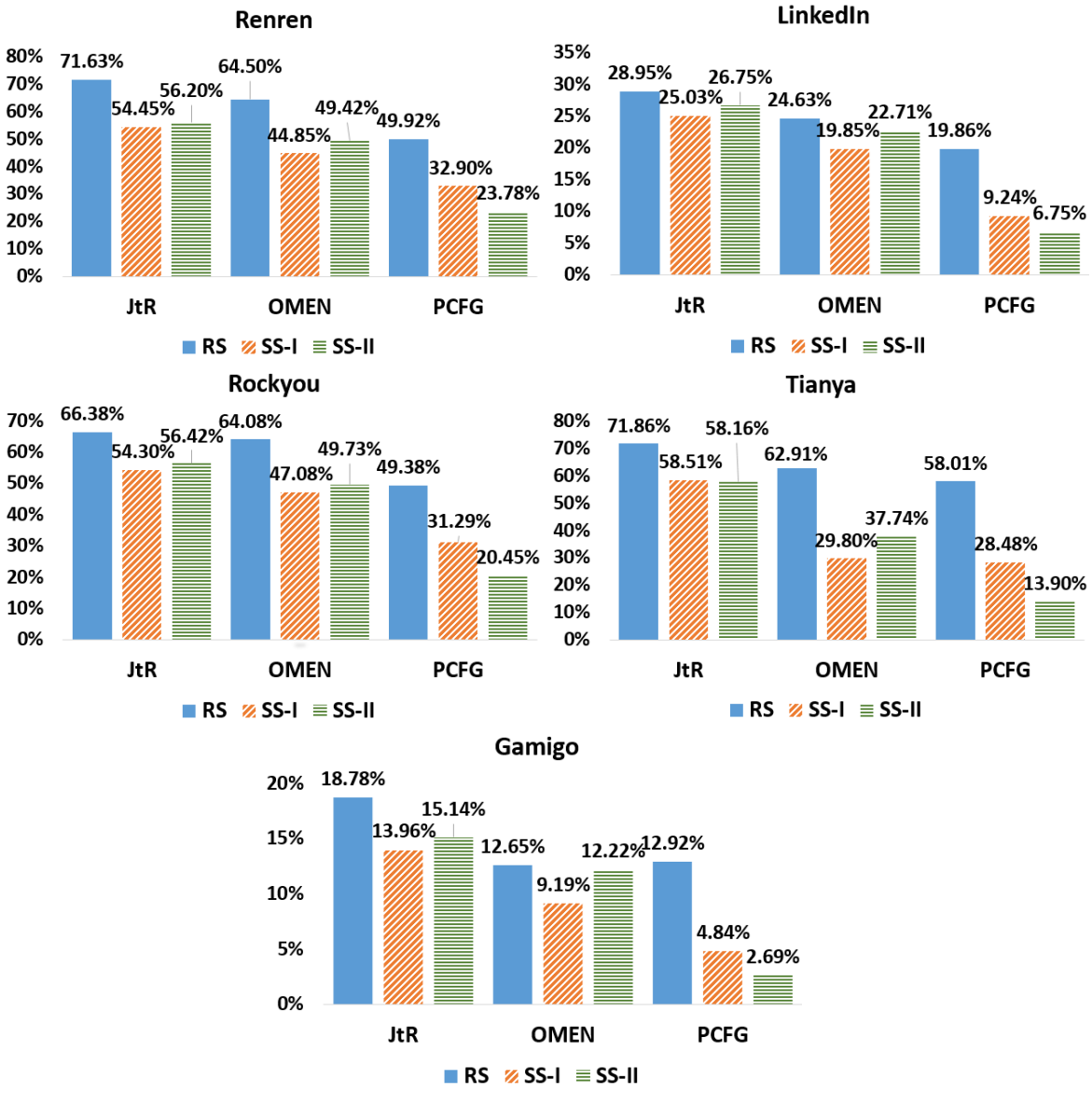


Figure 7: Diversity-based Cracking.

To evaluate the robustness of our metric, we look at both its effectiveness in protecting passwords from cracking models, and its possibility to leak important password distribution information like existing commercial strength checkers discussed in Chapter 3.

5.4.1 Attacking without Metric Details

In Figure 6, we describe the attack model to test if our proposed diversity-based measure can protect password datasets. We randomly select 50% of passwords from datasets shown in Table 1 and use them to train the cracking algorithms. From the other half of the passwords, we construct a password graph and run the Louvain Method to detect communities. Based on the number of communities and sizes of communities, we randomly select a fixed number of passwords from each of the communities as the selected samples. In our experiment setup, in each dataset we detect 5 communities. To make the selected sample size non-trivial, we randomly select 20000 passwords from each of the communities and thus obtaining 100000 passwords in each selected sample.

Finally, from the same password data we use to build the graph model, we randomly select 100000 passwords to form the random samples. Therefore, we obtain a selected sample that is based on our diversity-based metric, and a random sample that has statistically the same password distribution with the original dataset. We crack these two samples separately, with 10 billion guesses.

In the left part of Table 14, we show the diversity scores computed with the diversity metric of the 5 datasets. Ranked by the scores in ascending order, we see that Renren has the lowest diversity score while Gamigo has the highest. This suggests that Gamigo has a relatively more uniform distribution than other datasets and Renren has the most unbalanced distribution.

In Figure 7, we show the cracking results of the attack model. RS denotes the

random samples and SS-I denotes the selected sample from the password communities. For all datasets, the random samples have more cracked passwords than the selected sample. The selected sample is formed with regards to the diversity metric which aims to eliminate bias on the dataset and make the password distribution more uniform. Therefore, the selected sample is less vulnerable to algorithms trained with biased password distribution.

Furthermore, we see in Figure 6 that the general cracking rates for the datasets tend to follow an order of the ranking in Table 14. Renren, Tianya, and Rockyou have the highest cracking rates by JtR at 71.63%, 71.86%, and 66.38%, respectively, which are drastically higher than that in LinkedIn and Gamigo. This is consistent to statistics shown in Table 14 where Renren, Tianya, and Rockyou have similar diversity scores that are the lowest. Gamigo, having the highest diversity score, does have the lowest cracking rates with regards to all the cracking algorithms. Therefore, Figure 6 and Table 14 show consistent results, which means our diversity-based metric can provide effective and accurate indication on the security of passwords dataset.

5.4.2 Attacking with Metric Details

To examine if our metric has the same limitations as password strength checkers in Chapter 3, we conduct the same evaluation illustrated in Figure 1. We use the training data in Section 5.4.1 as *nonselective training* and the selected sample (SS-I in Figure 3) as the testing data. Assuming the attacker can obtain complete details of our metric including the weight values, and apply it to select training data by detection communities, we draw random samples from each of the communities to build the *Selective Training* dataset in the same way as we build the selected sample in Section 5.4.1. In this set up, we ensure the cracking evaluation results are comparable and we place them in the same figure denoted as SS-II.

From Figure 6, we see that PCFG has better cracking performance consistently

when trained with *nonselective training*, which implies the *selective training* does not introduce a similar structural distribution to the cracker. However, it is interesting that OMEN shows the exact opposite case with better performance when trained with *nonselective training*. JtR on the other hand, does not break the tie consistently either. The cracking rates of each scenario are generally very close with the *Nonselective* and *Selective* training, which is different from the obvious contrast observed in password strength checkers in Figure 2. The phenomenon shows that *selective training* in this case, may or may not enhance training-based cracking attack, and does not reveal useful password distribution to the adversary. This is reasonable because our diversity metric aims to balance the password distribution and eliminate possible bias. Therefore, it is not of typical interest for the adversary to leverage our diversity metric to enhance cracking.

In Table 16 and Table 17, we show the full results of cracking analysis conducted in Section 5.4.1 and Section 5.4.2, respectively.

Table 16: Cross-site Diversity-based Cracking

Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
JtR	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	18.78%	13.96%	20.68%	0.00%	17.13%	15.74%	20.60%	0.00%	12.98%	0.00%
LinkedIn	25.29%	21.50%	28.95%	25.03%	21.64%	22.61%	26.86%	25.74%	15.84%	19.70%
Renren	59.11%	44.48%	64.78%	48.66%	71.63%	54.45%	67.40%	52.40%	68.61%	49.14%
Rockyou	59.72%	43.49%	63.31%	47.15%	57.80%	45.76%	66.38%	54.30%	47.48%	38.23%
Tianya	56.68%	28.16%	61.58%	30.13%	68.83%	34.59%	63.92%	31.15%	71.86%	58.51%
Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
OMEN	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	12.65%	9.19%	17.81%	12.79%	11.18%	11.91%	18.08%	14.16%	5.16%	9.75%
LinkedIn	12.83%	12.19%	24.63%	19.85%	12.86%	16.91%	22.61%	20.86%	5.12%	14.00%
Renren	42.37%	26.14%	55.82%	35.34%	64.50%	44.85%	61.65%	40.63%	53.75%	33.52%
Rockyou	42.42%	28.23%	55.44%	36.46%	43.66%	32.93%	64.08%	47.08%	20.11%	23.61%
Tianya	44.53%	19.55%	57.43%	25.85%	65.76%	31.17%	60.89%	27.69%	62.91%	29.80%
Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
PCFG	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	12.92%	4.84%	13.42%	5.66%	13.79%	6.77%	15.05%	8.81%	13.44%	8.77%
LinkedIn	17.93%	8.50%	19.86%	9.24%	19.30%	11.07%	20.86%	13.65%	18.34%	14.37%
Renren	29.94%	20.53%	30.75%	20.95%	49.92%	32.90%	44.58%	29.59%	52.45%	34.13%
Rockyou	42.89%	23.04%	41.01%	22.01%	44.76%	25.17%	49.38%	31.29%	43.28%	28.11%
Tianya	25.01%	13.79%	28.08%	15.37%	42.80%	22.15%	41.28%	20.95%	58.01%	28.48%

Table 17: Selection Attack

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	15.14%	15.44%	15.98%	16.98%	14.41%
LinkedIn	23.53%	26.75%	25.39%	27.34%	21.71%
Renren	46.54%	49.77%	56.20%	52.90%	50.48%
Rockyou	44.17%	50.75%	50.65%	56.42%	42.83%
Tianya	30.12%	29.51%	57.50%	31.21%	58.16%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	12.22%	13.14%	11.95%	13.70%	9.17%
LinkedIn	15.95%	22.71%	18.19%	20.99%	13.51%
Renren	31.76%	38.24%	49.42%	41.84%	36.08%
Rockyou	28.79%	41.67%	39.67%	49.73%	24.95%
Tianya	24.80%	26.31%	31.48%	26.94%	37.74%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	2.69%	3.66%	3.55%	3.99%	3.06%
LinkedIn	4.53%	6.75%	5.97%	6.88%	5.11%
Renren	14.09%	16.83%	23.78%	19.43%	18.95%
Rockyou	15.36%	17.37%	19.01%	20.45%	16.64%
Tianya	9.21%	11.27%	13.59%	12.19%	13.90%

5.4.3 Attack on Passwords from User-study

Since the password diversity metric is used as an underlying implementation of the *exploitation* mode in *DPPG*, it is interesting to test the metric and the passwords created with *DPPG* in the same experiment. In Table 15, we show the results of using the *selective training* in Section 5.4.2 to crack the Mturk dataset in Section 4.3, which is comparable to the partial results in Table 11. We observe the same inconsistencies again in the results of both tables. Further, in the right part of Table 14, we show that the Mturk dataset has a higher *DivScore* than other sample datasets. When trained with random samples from original password datasets, PCFG can crack more passwords consistently of Mturk dataset and OMEN shows the opposite. The performance gain of OMEN in all scenarios are noticeable but insignificant. Therefore, passwords created with *DPPG* do not share common distribution with other passwords created using the similar diversity-based algorithm and thus are relatively secure even if the diversity metric is obtained by the adversary.

Remarks. In this section, to explore the security of password characteristics distribution, we define password diversity. To the best of our knowledge, this is the first attempt to quantify the diversity of passwords using various password attributes.

Based on password diversity, we propose a useful password security metric to evaluate the password dataset security. Our metric is different from traditional max-likelihood or min-entropy metrics which depends on specific rules that relate to individual password strength. Instead, the metric focuses more on the security of the password distribution contributed by each individual password. Through several attack-based evaluations, we show that the diversity metric while improving the security of password dataset, does not leak crucial information that significantly helps the attacker. The diversity-based metric also serves as a key component in *DPPG* in Chapter 4. Although *DPPG* uses the policy history to approximate password distribution, it still maintains accurate metadata with the global frequency map and cracking evaluation on the Mturk dataset is consistent with our assumptions.

CHAPTER VI

CONCLUSION

In this thesis, we study the password space and distribution to understand password dataset security better. Due to the limitation of existing strength measuring mechanisms, we propose a new and usable alternative based on an effective diversity metric to better protect passwords from offline cracking attacks.

We start by identifying issues with the existing commercial password strength checkers and evaluate them from the adversarial perspective. While previous work has analyzed the consistency and accuracy of the checkers, much effort has not been spent on their limitations of biasing and leaking password distributions to the adversary. Through our evaluation, we find that password strength checkers are effective in helping attackers mount more powerful attacks. The reason is that password strength checkers rely on static scoring policies that exert bias on the password distribution. The checkers can be leveraged by the attackers easily to select training data that are similar to the target passwords.

To propose an effective alternative that addresses the limitations of password strength checkers, we implement *DPPG* to generate dynamic policies for users, which is based on a password diversity metric and the current password distribution. To the best of our knowledge, *DPPG* is the first dynamic password policy generator that provides unpredictable dynamic policies and enforces usability control. Through *exploration* and *exploitation* modes, *DPPG* can expand the password space and balance password characteristics distribution which increase the overall security of the password dataset. Through a usability study, we test *DPPG* in practice and collect passwords for further analysis. Experiments are also conducted to show that the

collected passwords are more diverse in their attributes and have good security.

To study the password distribution and its security impact, we define the concept of password diversity. To the best of our knowledge, this is the first attempt to define and quantify password diversity considering a vector of password attributes. The quantification is extensible and can be adjusted with different weight values to shift the focus of measurement. To provide a way to analyze the password diversity of a dataset, we propose the diversity-based password security metric which is a key component for *DPPG* to generate effective policies. We also evaluate the metric from an adversarial perspective using it to sample data for an attack-based evaluation. Through cracking experiments in different setups, we conclude that the metric is effective in evaluating the security of password datasets and thus can serve as an effective start to evaluate password dataset security with regards to password diversity.

APPENDIX A

MECHANICAL TURK USER STUDY DOCUMENTS

In this appendix, we present documents relevant to our user study conducted on Amazon Mechanical Turk. Screenshots of the registration and login pages are also included.

Recruitment Statement.

We are conducting an experiment which requires you to create and maintain two user accounts (username and password) on two different websites. The websites are made up only to help you distinguish the two accounts. By going to the access link below, you will start a 4-session experiment with us. Each session is seperated from another by a time interval. At the end of each session, we will let you know what time you should return for the next session. Upon completing the last session, you will receive a pay code. Please copy and paste it into the box below to receive payment for participation. The entire process has a span of 6-7 days but your in-session participation time will only be around 12 minutes in total, because we ask you to come back to the sessions after some time. Compensation is only offered after you completed all 4 sessions in time.

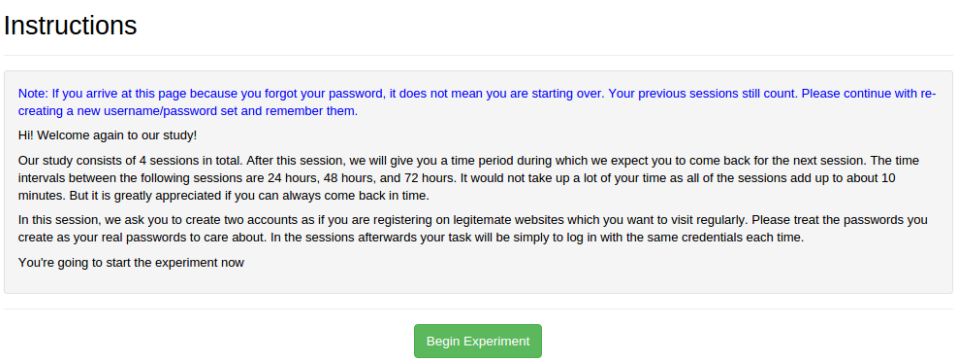


Figure 8: Instructions.

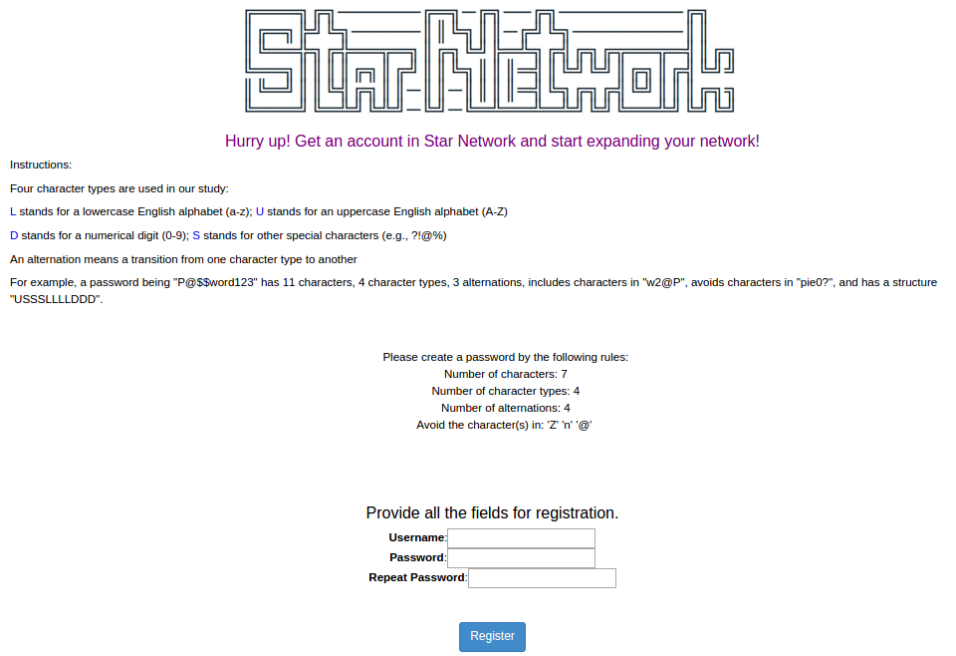


Figure 9: Registration with DPPG.

Figure 10: Registration with QQ's checker.

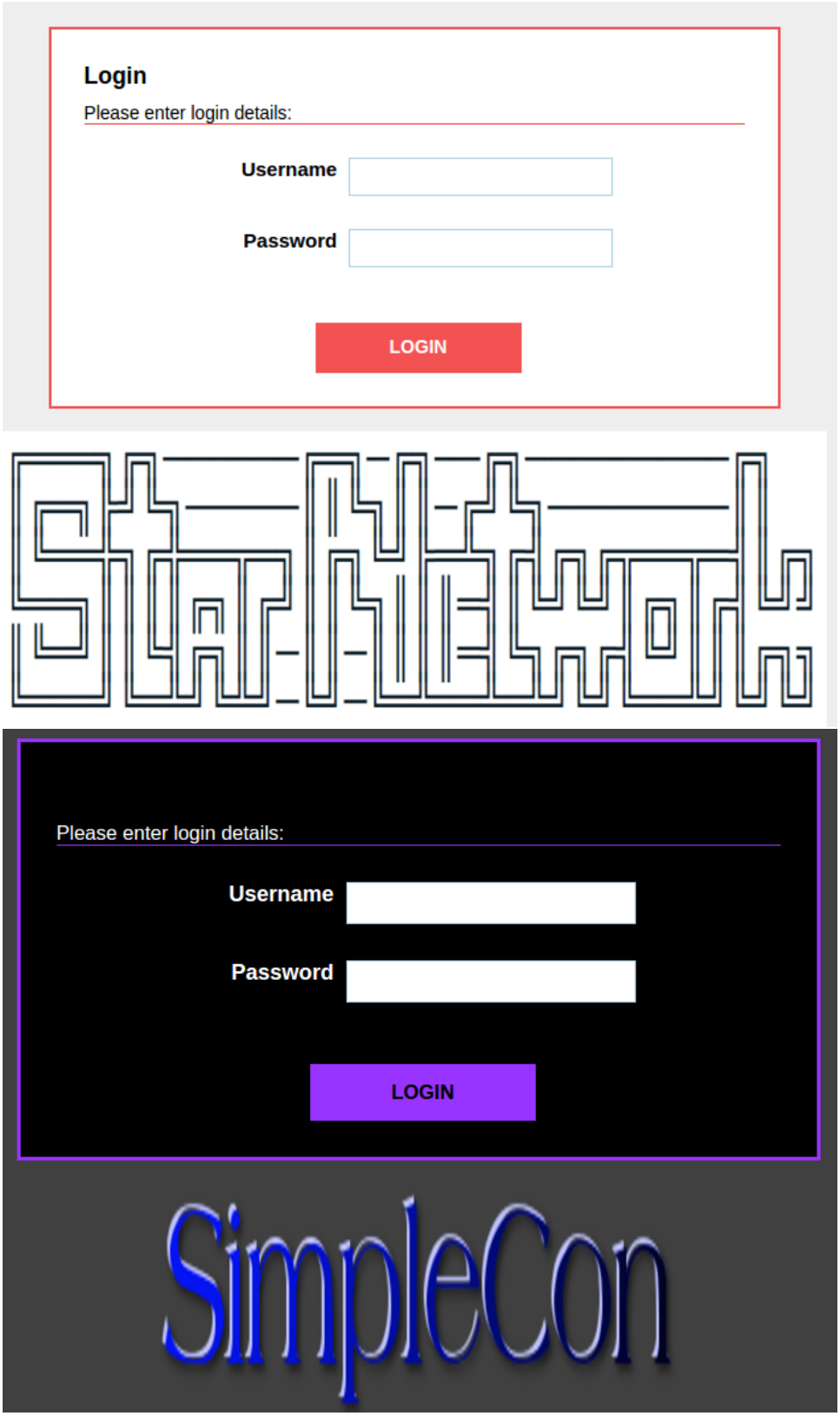


Figure 11: Login Pages.

REFERENCES

[1] "Amazon mechanical turk," *https://www.mturk.com/.*

[2] "Diversity-based password security metric and policy generation," *Anonymous for now.*

[3] "Gmail password leakage," *http://lifehacker.com/5-million-gmail-passwords-leaked-check-yours-now-1632983265.*

[4] "Hashcat v0.47," *http://hashcat.net/hashcat/.*

[5] "http://www.adeptus-mechanicus.com/codex/jtrhcmkv/jtrhcmkv.php,"

[6] "http://www.darkreading.com/attacks-and-breaches/yahoo-hack-leaks-453000-voice-passwords/d/d-id/1105289?,"

[7] "http://www.zdnet.com/blog/security/chinese-hacker-arrested-for-leaking-6-million-logins/11064,"

[8] "John the ripper 1.7.9-jumbo-7," *http://www.openwall.com/john/.*

[9] "John the ripper-bleeding-jumbo," *https://github.com/magnumripper/JohnTheRipper.*

[10] "Yahoo! password leakege," *http://www.cnet.com/news/yahoos-password-leak-what-you-need-to-know-faq/.*

[11] Abdalla, M., Bresson, E., Chevassut, O., Möller, B., and Pointcheval, D., "Provably secure password-based authentication in tls," *ASIACCS*, 2006.

[12] Akpulat, M., Bicakci, K., and Cil, U., "Revisiting graphical passwords for augmenting, not replacing, text passwords," *ACSAC*, 2013.

[13] Amico, M. D., Michiardi, P., and Roudier, Y., "Password strength: An empirical analysis," *Infocom*, 2010.

[14] Argyros, G. and Kiayias, A., "I forgot your password: Randomness attacks against php applications," *USENIX*, 2012.

[15] Blondel, V., Guillaume, J., Lambiotte, R., and Lefebvre, E., "Fast unfolding of communities in large networks," *Statistical Mechanics: Theory and Experiment*, 2008.

[16] Bojinov, H., Bursztein, E., Boyen, X., and Boneh, D., "Kamouflage: Loss-resistant password management," *ESORICS*, 2010.

[17] Bonneau, J., "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," *S&P*, 2012.

[18] Bonneau, J., "Guessing human-chosen secrets," *Doctoral*, Dissertation.

[19] Bonneau, J., Herley, C., Oorschot, P. C., and Stajano, F., "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," *S&P*, 2012.

[20] Bonneau, J. and Schechter, S., "Towards reliable storage of 56-bit secrets in human memory," *USENIX*, 2014.

[21] Burr, W. E., Dodson, D. F., and Polk, W. T., "Electronic authentication guideline," *NIST*, 2006.

[22] Camenisch, J., Lysyanskaya, A., and Neven, G., "Practical yet universally composable two-server password-authenticated secret sharing," *CCS*, 2012.

[23] Carnavalet, X. C. and Mannan, M., "From very weak to very strong: Analyzing password-strength meters," *NDSS*, 2014.

[24] Castelluccia, C., Dürmuth, M., and Perito, D., "Adaptive passwords-strength meters from markov models," *NDSS*, 2012.

[25] Chiasson, S., Biddle, R., and Oorschot, P. C. V., "A second look at the usability of click-based graphical passwords," *SOUPS*, 2007.

[26] Chiasson, S., Forget, A., Stobert, E., Oorschot, P. C. V., and Biddle, R., "Multiple password interference in text passwords and click-based graphical passwords," *CCS*, 2009.

[27] Chiasson, S., Oorschot, P. C. V., and Biddle, R., "A usability study and critique of two password managers," *USENIX*, 2006.

[28] Cox, L. P., Gilbert, P., Lawler, G., Pistol, V., Razeen, A., Wu, B., and Cheemalapati, S., "Spandex: Secure password tracking for android," *USENIX*, 2014.

[29] Das, A., Bonneau, J., Caesar, M., Borisov, N., and Wang, X., "The tangled web of password reuse," *NDSS*, 2014.

[30] Dirik, A. E., Memon, N., and Birget, J.-C., "Modeling user choice in the passpoints graphical password scheme," *SOUPS*, 2007.

[31] Dunphy, P., Heiner, A. P., and Asokan, N., "A closer look at recognition-based grpahical passwords on mobile devices," *SOUPS*, 10.

[32] Dunphy, P., Nicholson, J., and Olivier, P., "Securing passfaces for description," *SOUPS*, 2008.

[33] Dunphy, P. and Olivier, P., "On automated image choice for secure and usable graphical passwords," *ACSAC*, 2012.

[34] Dürmuth, M., Chaabane, A., Perito, D., and Castelluccia, C., "When privacy meets security: Leveraging personal information for password cracking," *CoRR abs/1304.6584*, 2013.

[35] Egelman, S., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., and Herley, C., "Does my password go up to eleven? the impact of password meters on password selection," *CHI*, 2013.

[36] Fahl, S., Harbach, M., Acar, Y., and Smith, M., "On the ecological validity of a password study," *SOUPS*, 2013.

[37] Florêncio, D. and Herley, C., "Klassp: Entering passwords on a spyware infected machine using a shared-secret proxy," *ACSAC*, 2006.

[38] Florêncio, D. and Herley, C., "A large-scale study of web password habits," *WWW*, 2007.

[39] Florêncio, D. and Herley, C., "Where do security policies come from," *SOUPS*, 2010.

[40] Florêncio, D., Herley, C., and Coskun, B., "Do strong web passwords accomplish anything," *HotSec*, 2007.

[41] Florêncio, D., Herley, C., and Oorschot, P. C. V., "Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts," *USENIX*, 2014.

[42] Forget, A., Chiasson, S., Oorschot, P. C. V., and Biddle, R., "Improving text passwords through persuasion," *SOUPS*, 2008.

[43] Gao, H., Guo, X., Chen, X., Wang, L., and Liu, X., "Yagp: Yet another graphical password strategy," *ACSAC*, 2008.

[44] Gasti, P. and Rasmussen, K. B., "On the security of password manager database formats," *ESORICS*, 2012.

[45] Gaw, S. and Felten, E. W., "Password management strategies for online accounts," *SOUPS*, 2006.

[46] Golofit, K., "Click passwords under investigation," *ESORICS*, 2007.

[47] Haque, S. M. T., Scielzo, S., and Wright, M., "Applying psychometrics to measure user comfort when constructing a strong password," *SOUPS*, 2014.

[48] Hart, M., Castille, C., Harpalani, M., Toohill, J., and Johnson, R., "Phorcefield: A phish-proof password ceremony," *ACSAC*, 2011.

[49] Herley, C., Oorschot, P. C., and Patrick, A. S., "Passwords: If we're so smart, why are we still using them?," *FC*, 2009.

[50] Hlywa, M., Biddle, R., and Patrick, A. S., "Facing the facts about image type in recognition-based graphical passwords," *ACSAC*, 2011.

[51] Horst, T. W. and Seamons, K. E., "Simple authentication for the web," *SECURECOMM*, 2007.

[52] Horst, T. W. and Seamons, K. E., "pwdarmor: Protecting conventional password-based authentications," *ACSAC*, 2008.

[53] Houshmand, S. and Aggarwal, S., "Building better passwords using probabilistic techniques," *ACSAC*, 2012.

[54] Jeyaraman, S. and Topkara, U., "Have the cake and eat it too - infusing usability into text-password based authentication systems," *ACSAC*, 2005.

[55] Ji, S., Yang, S., and Beyah, R., "Pars: A uniform and open-source password analysis and research system," *ACSAC*, 2015.

[56] Ji., S., Yang, S., Hu, X., Han, W., Li, Z., and Beyah, R., "Zero-sum password cracking game: A large-scale empirical study on the crackability, correlation, and security of passwords," *Dependable and Secure Computing, IEEE Transactions on*, 2015.

[57] Juels, A. and Rivest, R. L., "Honeywords: Making password-cracking detectable," *CCS*, 2013.

[58] Just, M. and Aspinall, D., "Personal choice and challenge questions: A security and usability assessment," *SOUPS*, 2009.

[59] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and López, J., "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," *S&P*, 2012.

[60] Keyboard_dic, "https://sites.google.com/site/reusablesec/home/custom-wordlists,"

[61] Kim, T. H.-J., Stuart, H. C., Hsiao, H.-C., Lin, Y.-H., Zhang, L., Dabbish, L., and Kiesler, S., "Yourpassword: Applying feedback loops to improve security behavior of managing multiple passwords," *ASIACCS*, 2014.

[62] Komanduri, S., R.Shay, Cranor, L. F., Herley, C., and Schechte, S., "Telepathwords: Preventing weak passwords by reading users' minds," *USENIX*, 2014.

[63] Kontaxis, G., Athanasopoulos, E., Portokalidis, G., and Keromytis, A. D., "Sauth: Protecting user accounts from password database leaks," *CCS*, 2013.

[64] Kumar, M., Garfinkel, T., Boneh, D., and Winograd, T., "Reducing shoulder-surfing by using gaze-based password entry," *SOUPS*, 2007.

[65] Kuo, C., Romanosky, S., and Cranor, L. F., "Human selection of mnemonic phrase-based passwords," *SOUPS*, 2006.

[66] Li, Z., Han, W., and Xu, W., "A large-scale empirical analysis on chinese web passwords," *Usenix Security*, 2014.

[67] Li, Z., He, W., Akhawe, D., and Song, D., "The emperor's new password manager: Security analysis of web-based password managers," *USENIX*, 2014.

[68] Liu, D., Cuervo, E., Pistol, V., Scudellari, R., and Cox, L. P., "Screenpass: Secure password entry on touchscreen devices," *MobiSys*, 2013.

[69] Luca, A. D., Denzel, M., and Hussmann, H., "Look into my eyes! can you guess my password?," *SOUPS*, 2009.

[70] Ma, J., Yang, W., Luo, M., and Li, N., "A study of probilistic password models," *S&P*, 2014.

[71] Mazurek, M. L., Komanduri, S., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., Kelley, P. G., Shay, R., and Ur, B., "Measuring password guessability for an entire university," *CCS*, 2013.

[72] McCarney, D., Barrera, D., Clark, J., Chiasson, S., and Oorschot, P. C. V., "Tapas: Design, implementation, and usability evaluation of a password manager," *ACSAC*, 2012.

[73] McCune, J. M., Perrig, A., and Reiter, M. K., "Safe passage for passwords and other sensitive data," *NDSS*, 2009.

[74] Molloy, I. and Li, N., "Attack on the gridcode one-time password," *ASIACCS*, 2011.

[75] Narayanan, A. and Shmatikov, V., "Fast dictionary attacks on passwords using time-space tradeoff," *CCS*, 2005.

[76] Polakis, I., Lancini, M., Kontaxis, G., Maggi, F., Ioannidis, S., Keromytis, A. D., and Zanero, S., "All your face are belong to us: Breaking facebook's social authentication," *ACSAC*, 2012.

[77] Rabkin, A., "Personal knowledge questions for fallback authentication: Security questions in the era of facebook," *SOUPS*, 2008.

[78] Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C., "Stronger password authentication using browser extensions," *USENIX*, 2005.

[79] Salehi-Abari, A., Thorpe, J., and Oorschot, P. C. V., "On purely automated attacks and click-based graphical passwords," *ACSAC*, 2008.

[80] Sayegh, A. A. and El-Hadidi, M. T., "A modified secure remote password (srp) protocol for key initialization and exchange in bluetooth systems," *SECURECOMM*, 2005.

[81] Schaub, F., Walch, M., Könings, B., and Weber, M., "Exploring the design space of graphical passwords on smartphones," *SOUPS*, 2013.

[82] Schechter, S., Brush, A. J. B., and Egelman, S., "It's no secret: Measuring the security and reliability of authentication via 'secret' questions," *S&P*, 2009.

[83] Schechter, S., Herley, C., and Mitzenmacher, M., "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," *USENIX HotSec'10*, 2010.

[84] Schechter, S., Herley, C., and Mitzenmacher, M., "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," *HotSec*, 2010.

[85] Schechter, S. and Reeder, R. W., "1 + 1 = you: Measuring the comprehensibility of metaphors for configuring backup authentication," *SOUPS*, 2009.

[86] Schechter, S. E., Dhamija, R., Ozment, A., and Fischer, I., "The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies," *S&P*, 2007.

[87] Schmidt, D. and Jaeger, T., "Pitfalls in the automated strenghting of passwords," *ACSAC*, 2013.

[88] Shay, R., Kelley, P. G., Komanduri, S., Mazurek, M. L., Ur, B., Vidas, T., Bauer, L., Christin, N., and Cranor, L. F., "Correct horse battery staple: Exploring the usability of system-assigned passphrases," *SOUPS*, 2012.

[89] Shay, R., Komanduri, S., Kelley, P. G., Leon, P. G., Mazurek, M. L., Bauer, L., Christin, N., and Crano, L. F., "Encountering stronger password requirements: User attitudes and behaviors," *SOUPS*, 2010.

[90] Shirvanian, M., Jarecki, S., Saxena, N., and Nathan, N., "Two-factor authentication resilient to server compromise using mix-bandwidth devices," *NDSS*, 2014.

[91] Silver, D., Jana, S., Boneh, D., Chen, E., and Jackson, C., "Password managers: Attacks and defenses," *USENIX*, 2014.

[92] Stobert, E. and Biddle, R., "Memory retrieval and graphical passwords," *SOUPS*, 2013.

[93] Stobert, E. and Biddle, R., "The password life cycle: User behaviour in managing passwords," *SOUPS*, 2014.

[94] Stobert, E., Forget, A., and Chiasson, S., "Exploring usability effects of increasing security in click-based graphical passwords," *ACSAC*, 2010.

[95] Stock, B. and Johns, M., "Protecting users against xss-based password manager abuse," *ASIACCS*, 2014.

[96] Sun, H.-M., Chen, Y.-H., Fang, C.-C., and Chang, S.-Y., "Passmap: A map based graphical-password authentication system," *ASIACCS*, 2012.

[97] Suo, X., Zhu, Y., and Owen, G. S., "Graphical passwords: A survey," *ACSAC*, 2005.

[98] Tannous, A., Trostle, J., Hassan, M., McLaughlin, S. E., and Jaeger, T., "New side channels targeted at passwords," *ACSAC*, 2008.

[99] Tari, F., Ozok, A. A., and Holden, S. H., "A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords," *SOUPS*, 2006.

[100] Thorpe, J., MacRae, B., and Salehi-Abari, A., "Usability and security evaluation of geopass: a geographic location-password scheme," *SOUPS*, 2013.

[101] Thorpe, J. and Oorschot, P. C. V., "Human-seeded attacks and expliting hot-spots in graphical passwords," *USENIX*, 2007.

[102] Tian, J., Qu, C., Xu, W., and Wang, S., "Kinwrite: Handwriting-based authentication using kinect," *NDSS*, 2013.

[103] Uellenbeck, S., Dürmuth, M., Wolf, C., and Holz, T., "Quantifying the security of graphical passwords: The case of android unlock patterns," *CCS*, 2013.

[104] Ur, B., Kelley, P. G., Komanduri, S., Lee, J., Maass, M., Mazurek, M. L., Passaro, T., Shay, R., Vidas, T., Bauer, L., Christin, N., and Cranor, L. F., "How does your password measure up? the effect of strength meters on password creation," *USENIX*, 2012.

[105] Veras, R., Collins, C., and Thorpe, J., "On the semantic patterns of passwords and their security impact," *NDSS*, 2014.

[106] Wang, L., Ohta, K., and Kunihiro, N., "Password recovery attack on authentication protocol md4(password —— challenge)," *ASIACCS*, 2008.

[107] Wang, Z., Jing, J., and Li, L., "Time evolving graphical password for securing mobile devices," *ASIACCS*, 2013.

[108] Weir, M., Aggarwal, S., Collins, M., and Stern, H., "Testing metrics for password creation policies by attacking large sets of revealed passwords," *CCS*, 2010.

[109] Weir, M., Aggarwal, S., Medeiros, B., and Glodek, B., "Password cracking using probabilistic context-free grammars," *S&P*, 2009.

[110] Wiedenbeck, S., Birget, J.-C., and Brodskiy, A., "Authentication using graphical passwords: Effects of tolerance and image choice," *SOUPS*, 2005.

[111] Wimberly, H. and Liebrock, L. M., "Using fingerprint authentication to reduce system security: An empirical study," *S&P*, 2011.

[112] Wright, N., Patrick, A. S., and Biddle, R., "Do you see your password? applying recognition to textual passwords," *SOUPS*, 2013.

[113] Yan, J., Blackwell, A., and Anderson, R., "Password memorability and security: Empirical results," *S&P*, 2004.

[114] Yan, Q., Han, J., Li, Y., Zhou, J., and Deng, R. H., "Designing leakage-resilient password entry on touchscreen mobile devices," *ASIACCS*, 2013.

[115] Yang, Y., Deng, R. H., and Bao, F., "Fortifying password authentication in integrated healthcare delivery systems," *ASIACCS*, 2006.

[116] Yazdi, S. H., "Analyzing password strength and efficient password cracking," *Theses,*, Treatise.

[117] Yee, K.-P. and Sitaker, K., "Passpet: Convenient password management and phishing protection," *SOUPS*, 2006.

[118] Zakaria, N. H., Griffiths, D., Brostoff, S., and Yan, J., "Shoulder surfing defence for recall-based graphical passwords," *SOUPS*, 2011.

[119] Zhang, Y., Monrose, F., and Reiter, M. K., "The security of modern password expiration: An algorithmic framework and empirical analysis," *CCS*, 2010.