An In-Depth Analysis of Guesser Behaviour

by

Connor Cushing

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Computer Science

University of Ontario Institute of Technology (Ontario Tech University)

Supervisor: Dr. Julie Thorpe and Dr. Amirali Abari

November 2020

Thesis Examination Information

Submitted by: Connor Cushing

Master of Science in Computer Science

Thesis title: An In-Depth Analysis of Guesser Behaviour

An oral defense of this thesis took place on October 7, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee: Dr. Faisal Qureshi

Research Supervisor: Dr. Julie Thorpe

Research Co-supervisor: Dr. Amirali Salehi-Abari

Examining Committee Member: Dr. Ying Zhu

Thesis Examiner: Dr. Ken Pu

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

We propose a methodology to perform an in-depth analysis on different password guessers and their guessing abilities. We devise new metrics and statistics that directly compare the types of passwords each guesser generates, extending analysis beyond number of passwords guessed which is the primary form of analysis in literature currently. This approach allows for a fine-grained analysis where we compare the guesses produced by each guesser when trained on varied real-world datasets and under different conditions (e.g., limited training data, limited number of guesses, or dissimilar training and testing data). We find that similarity of training to testing data is more important than dataset size, and that some guessers are better equipped to deal with dissimilarity than others. We demonstrate that guessers often produce dissimilar guesses, even when trained on the same training data. This result is leveraged to show how guessers with lower resource requirements can be combined to guess a comparable number of passwords as more resource intensive tools. Our methodology can be applied in the future to better compare new guessing tools, and our insights allow us to provide concrete advice for systems administrators performing reactive checking with modern tools.

Keywords: passwords; computer security; user authentication

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____ Connor Cushing

Statement of Contributions

Part of the work described in Chapter 5 has appeared in:

Z. Parish, C. Cushing, S. Aggarwal, A. Salehi-Abari, J. Thorpe, "Password Guessers Under a Microscope: An In-Depth Analysis to Inform Deployments", *arXiv preprint arXiv: 2008.07986* (2020). [69]

I performed the majority of the creation of the framework, analysis of the guessers, and writing of the manuscript.

Acknowledgements

A number of people were integral in both my research and the writing of this thesis including my supervisors, Dr. Julie Thorpe and Dr. Amirali Abari; my committee members, and my family and friends.

I would like to begin by thanking my supervisors, Dr. Julie Thorpe and Dr. Amirali Abari, for their patience and support throughout my research. They both supported and guided me throughout my degree and provided me more opportunities than I could ever ask for.

I would also like to give a special thanks to my supervisory committee member Dr. Ying Zhu, and my external examiner Dr. Ken Pu, for their valuable input throughout this thesis editing process.

I want to thank my parents for their eternal belief in me. While they may not fully understand what I do, their pride in me always has and will continue to empower me to pursue my goals in life.

I would like to thank Zachary Parish for his valuable assistance in the lab, both as an assistant in the experiments as well as a friend.

Last, but certainly not least, I would like to thank both Jacob Drummond and Christopher Corradini for their support, acceptance, and care for me during the course of this degree. They significantly supported both my mental and physical health during this process and without them I would have never made it this far.

Contents

Thesis Examination Information i

Abstract ii

Author's Declaration iii

Statement of Contributions iv

Acknowledgements v

Contents vi

List of Figures ix

List of Tables x

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Thesis Summary . . . 3
- 1.3 Thesis Statement . . . 4
- 1.4 Contributions . . . 4
- 1.5 Thesis Organization . . . 5

2 Related Works 6
- 2.1 Overview . . . 6
- 2.2 Weaknesses of Passwords . . . 7
- 2.3 Alternative Authentication . . . 14
- 2.4 Password Policies . . . 18
- 2.5 Password Meters . . . 21
- 2.6 Password Guessers . . . 25
- 2.7 Literature Gap . . . 28

3 Metrics and Algorithms 30
- 3.1 Password Features . . . 31

3.2 Statistics . . . 32
3.2.1 Cosine Similarity . . . 33
3.2.2 Jaccard Index . . . 34
3.2.3 Generalized Jaccard Index . . . 34
3.3 Guessing Success Statistics . . . 35
3.3.1 Success Rate . . . 36
3.3.2 Guesser Success Rate . . . 36
3.3.3 Training Success Rate . . . 37
3.3.4 Fixed Success Rate . . . 37
3.4 Guessing Behaviour Statistics . . . 37
3.4.1 Guessing Similarity . . . 37
3.4.2 Training Similarity . . . 38
3.4.3 Training Independence . . . 38

4 Experimental Methodology **40**
4.1 Comparison Resources . . . 40
4.1.1 Password Datasets . . . 41
4.1.2 Password Guessing Tools . . . 42
4.2 Guessing Passwords . . . 45
4.3 Resource Measurements of Password Guessers . . . 46
4.4 Password Guessing Success . . . 46
4.5 Password Guessing Behaviour . . . 48
4.6 Combining Guessers . . . 49

5 Experiments **51**
5.1 Resource Measurements of Password Guessers . . . 51
5.2 Password Guessing Performance . . . 52
5.2.1 Impact of Password Guesser Choice . . . 52
5.2.2 Impact of Training Data Choice . . . 54
5.2.3 Summary . . . 60
5.3 Password Guessing Behaviour . . . 60
5.3.1 Training Independence . . . 60
5.3.2 Generalizability . . . 61
5.3.3 Sensitivity to Training Dataset Size . . . 62
5.3.4 Guessing Similarity . . . 63
5.3.5 Successful Guessing Similarity . . . 65
5.4 Combining Guessers . . . 66
5.4.1 Individual Guessers . . . 67
5.4.2 Combination Attacks . . . 67

6 Discussion **71**

7 Conclusions and Future Work **74**

Bibliography **76**

List of Figures

5.1 The average success rates for various pairs of training and testing datasets. Each edge width is proportional to the average success rate of all guessers for a fixed training and testing dataset. The edge colors match the training dataset color and also the outer edges are directed clockwise from training to testing dataset. The node sizes represent the size of the password datasets. 55

5.2 Plaintext datasets with their pairwise (a) cosine similarity, (b) generalized Jaccard similarity, (c) cosine training similarity, and (d) Jaccard training similarity. The training similarity between datasets is computed by Eq. 3.12. The edge weights and colors are based on the corresponding metric value between two datasets. The node color captures the metric average for the corresponding dataset. The node size is proportional to the dataset size. 57

5.3 The cosine and Jaccard guessing similarity (see Eq. 3.10) between guessers at the cutoffs of 1 million or 300 million guesses. The edge colors represent the similarity value between two guessers. The edge width further highlights the relative similarities within a figure (thicker means more similar). The node size represents the guesser's average success rate. The node colors represent their average similarity. . . . 64

5.4 The generalized Jaccard guessing similarity between guessers for successful guesses. The edge weights and colors represent the similarity metric between two guessers. The size of the nodes represent the average success rate of the guesser. The node colors represent the average similarity of guessers with all other guessers. 65

5.5 Performance of guessers trained on the Merged dataset and tested against LinkedIn. The dotted line marks the Identity guesser's last guess at 67 million guesses, each other guesser made 2 billion guesses. 68

List of Tables

4.1 The password datasets, their sizes, and the ratio between unique and total number of passwords. *Merged contains all other plaintext datasets in this table. 41

5.1 Guesser training and generation time. Each dataset is of different size and randomly sampled from the Merged Dataset. Guessers have generated 300M guesses. 52
5.2 Guessers' mean success rates at 1 Million and 300 Million guesses (standard deviations in parenthesis). The two best and worst are highlighted with green and red, respectively. 53
5.3 Mean success rates for training password datasets. Datasets are ordered smallest to largest from left to right. 56
5.4 Guessers' Jaccard training independence. 61
5.5 Guessers' generalizability, with 300M guess cutoff. A higher success rate indicates a better ability to generalize. 62
5.6 The mean percentage of passwords guessed by each guesser when trained on different-sized subset of Twitter with a cutoff of 300M. 62
5.7 Percentage of LinkedIn passwords successfully guessed. Guessers are trained on the Merged dataset and cutoff at 2 billion guesses. 67
5.8 The percentage of LinkedIn passwords cracked by an offline attack using the Identity guesser followed by a combination of guessers, each making two billion guesses. The names of guessers are shortened to their first letters: (P)CFG, (O)MEN, (N)N, (S)em, and (J)tR-Markov. Each combination attack is color-coded by its sequential runtime for training and guess generation of its guessers: Green is less than 8 hours (i.e., a workday), yellow is less than 16 hours, and red is over two weeks. 69

Chapter 1

Introduction

To authorize legitimate access to resources or information, a computer system requires authentication to verify a user's identity. Typically, this verification is done with something the user knows (e.g., a text password, or PIN), something the user has (e.g., security tokens), or something the user is (e.g., retina scans or other biometrics). Studies have shown how text passwords are considered an insecure form of authentication [14, 31, 56, 64, 68, 85, 86, 90]. Research into alternate, more secure authentication systems has proven ineffective at replacing text passwords as a dominant form of authentication [15].

1.1 Motivation

Despite their popularity, text passwords—from here on referred to as passwords—suffer from many security issues. Given the importance and frequent use of passwords, the news of frequent password leaks [45] is even more alarming. Leaks may result from mistakenly giving passwords to attackers [78], password reuse across accounts [25], or online and offline attacks on authentication systems [14, 31, 56, 64, 68, 85, 86, 90].

While one can defend against online attacks by limiting the number of login attempts allowed, defending against offline attacks is more challenging. In offline attacks, an attacker steals a database of (usually) hashed passwords, and tries to recover (or "crack") the passwords through offline guessing. As this database is offline, it is out of our control, and there is no limit to the number of guess attempts an attacker can make. It is recommended that authentication systems use computationally expensive hash functions to slow down an attacker's guessing progress. Unfortunately, even with the most computationally expensive hash functions, many passwords can be easily guessed, as users often create predictable passwords. Furthermore, these leaked passwords can reveal information about unleaked passwords as they are often similar to each other [50]. Worse yet, password reuse between accounts is common [25], so compromising a password for one account can lead to the compromising of other accounts.

Despite decades of research into alternate, more secure, authentication systems, passwords remain a dominant form of authentication [15]. Text password systems often have an advantage in deployment as they are easy to implement and familiar to most users [15]. These advantages reduce the likelihood of passwords being replaced in the foreseeable future. To protect against password guessing attacks, administrators are advised to perform password checking, either *proactively* at the time of password creation or *reactively* through attempting to crack their own password databases [12]. These assessments can only be as accurate and powerful as the tools used to perform them. However, there is considerable uncertainty as to which password guessers to use, in a particular use case, as the only available data regarding password guessing behaviour is their success rate under disparate benchmarks. To make an informed decision, an administrator must understand how password guessers behave, compare to, and compliment each other under a variety of conditions. Unfortunately,

the literature lacks both an analytical framework for studying guessers' behaviours and studies involving a deep comparison of many guessers' behaviours under varied conditions.

1.2 Thesis Summary

Previous research has focused on developing stronger guessers and training datasets, with less emphasis on comparing and complementing each other. We propose an analytical framework to compare the behaviours of password guessers under varied settings. Using this framework, we analyze a number of well-studied and popular password guessing tools and training password datasets to reveal important practical insights and comparisons. Our analyses demonstrate that guessers offer various levels of generalizability, and their success rates are more closely related to the similarity between training and testing datasets than training dataset size. Our results also show that guessers often generate dissimilar guesses, which can be leveraged for more effective password checking. We show how administrators can get more bang for their buck by using combinations of computationally-cheap guessers that, when used together, compare to computationally-intensive guessers. A key take-away from our analysis is that the value of a password guesser is not only in its success rate, but also in its ability to compliment other guessers. Our findings allow us to provide concrete recommendations for system administrators performing password checking. Our analytical framework serves future password research as well as practitioners for understanding new guessing tools' behaviour, and their ability to compliment or substitute other existing guessers.

1.3 Thesis Statement

This thesis proposes an analytical framework for comparing password guessers. Under this framework, the thesis analyzes various state-of-the-art password guesses in various circumstances to discover how their behaviours change and compliment each other. Using its analyses, this thesis also explores how to combine low-resource password guessers to achieve comparable results of strong, high-resource guessers.

The following questions are addressed in this thesis:

1. How can one objectively compare password guessers beyond comparing successful guessing rates?

2. How do the behaviours of password guessers change when their training and testing data change?

3. How can multiple password guessers be used together effectively?

1.4 Contributions

The thesis contributions are as follows: (1) The design and implementation of a framework for in-depth analysis and comparison of different password guessers beyond the traditional method of only comparing guessing success rates. (2) A detailed analysis of password guesser behaviours under various training and testing datasets. (3) The design, implementation, and evaluation of a method for combining multiple password guessers together to compliment each other for increased guessing success rates. (4) Insight and guidance on how deploy the framework and analyses in practice for password checking.

1.5 Thesis Organization

- **Chapter 2: Related Works** presents previous work in password research. It focuses on the weaknesses of passwords, alternative authentication, password composition policies, password strength meters, password guessing techniques, and lastly discuss where this work fits in the literature.

- **Chapter 3: Metrics and Algorithms** describes the password summarization metrics and comparison algorithms used extensively throughout the work.

- **Chapter 4: Experimental Methodology** describes in detail the experiments used to compare, contrast, and combine password datasets and password guessing techniques.

- **Chapter 5: Experiments** presents and analyzes the results from the techniques described in Chapter 4.

- **Chapter 6: Discussion** explains recommendations for how the described techniques could be used in non-research settings.

- **Chapter 7: Future Work and Conclusions** presents a summary of the work and possible ways in which the research could be expanded.

Chapter 2

Related Works

2.1 Overview

Passwords are often the first line of defense in protecting our confidential information. Unfortunately, it has been repeatedly shown that user passwords are often similar or identical, and are consequently guessable by an adversary [14, 21, 61, 63]. With no alternative catching on currently, researchers have been attempting to improve the strength of passwords a number of different ways and demonstrate how easy current passwords are to guess.

Our literature review begins with an investigation of the weaknesses of passwords (see Section 2.2) to provide insight on the troubling situation we find ourselves in. We then explore work in alternative authentication techniques (see Section 2.3), their strengths, weaknesses, and why they haven't replaced text passwords as a dominant form of authentication. Following this, we explore two prominent fields of strengthening text passwords: password composition policies (see Section 2.4) and password strength meters (see Section 2.5). Lastly, we explore related work on password guessing tools (see Section 2.6) and discuss how our research fits in with previous work

(see Section 2.7).

2.2 Weaknesses of Passwords

Since the use of passwords for authentication is so wide-spread, one would expect passwords to be relatively secure. Unfortunately, passwords are widely regarded as insecure for a number of reasons including poor storage habits by administrators [45] and poor creation habits by users. We begin our literature review by exploring research on how and why users create weak passwords. Even if an administrator uses the most secure systems for password authentication, a weak password created by a user makes their account vulnerable.

While a completely random password may not be memorable for a user, users might employ many different techniques to increase the memorability of their password. Unfortunately, many of these techniques create noticeable patterns in their passwords which can be exploited. Some examples of these techniques include keyboard patterns in passwords [74] (e.g., "qwerty" or "1q2w3e4") or replacing letters with similar looking special characters [49]. The rampant use and weaknesses of keyboard patterns were studied by Schweitzer *et al.* [74]. While these patterns may appear random at a glance, they are easy for a user to remember. Using visualization techniques, the keyboard patterns were collected and categorized. This information was then shown to be effective in improving the success rate of guessing attacks.

By investigating five different datasets, Jakobsson *et al.* [49] discover how special characters (e.g., @, ?, etc.) are typically used in passwords. Typically, they found special characters were placed at the end of a password, but some were used according to simple patterns like "L33T". The "L33T" pattern is where similar looking symbols and numbers are substituted for letters (such as replacing 'a' with '@'). The most

common special characters were ones used in "L33T" (e.g. '@' or '&') whereas symbols not used commonly in "L33T" were rarely used (e.g. '?' or '}'). These special characters were still the most used even if a password wasn't written in "L33T".

Password re-use is also a rampantly used technique to remember passwords. A study of over 500 thousand users by Florencio *et al.* [34] investigated how prevalent this is. On average, users seem to have 7 unique passwords, but an unsettling discovery was how the average password is used for approximately 6 different websites. In general, stronger passwords were reused less than weaker passwords. These weaker passwords being used for more websites could have disastrous consequences, as these passwords are more vulnerable to attack at one location. Once a password is leaked from a website, an attack against logins at other websites with the same password becomes simple.

Users may also alter their passwords in predictable ways. While this creates a unique password, these passwords are typically simple to guess. A survey and analysis of leaked password datasets by Das *et al.* [25] found that 77% users either re-use or modify an existing password between logins on different websites. This information is leveraged to create a cross-site password guessing algorithm, which manages to guess 30% of a user's transformed passwords within 100 guesses, compared to the 14% guessed by a standard guessing algorithm.

A 50 person study was performed by Hanamsagar *et al.* [42] to determine reasoning behind password habits. By asking the participants to log into 12 actual accounts (where the passwords were semantically transformed for privacy reasons) the authors were given the opportunity to question the participants about their actual password habits. They found that people had 80 online accounts on average, a drastic increase from the previous estimate of 25 by Florenco *et al.* [34]. They also found that password reuse is rampant, even for accounts participants stated were important (i.e. bank

accounts). Participants typically claimed that they didn't reuse passwords or only shared passwords between unimportant accounts. Despite belief of the contrary, risk-averse participants did not produce stronger passwords. While many participants understand good password creation strategies, some still had weak passwords. Many participant's passwords had some strong characteristics, but were typically shorter than 10 characters.

Some users have also been known to use personal information, such as their name, birthday, or username, to improve the memorability of their password. Dürmuth *et al.* [32] investigated how including this information might be exploited by attackers. They start by calculating the Jaccard index and Longest Common Substring (LCSS) between passwords and personal information. While only a relatively small number of passwords seem to have personal information in them, first names, birthdays, and usernames are found to have the highest overlap with passwords. This information is then used to boost the abilities of a Markov-based password guesser, resulting in a 5% increase in passwords guessed at 100 million guesses with limited access to personal information.

Understanding languages used by users might also be leveraged by attackers. Li *et al.* [57] studied 100 million passwords, comparing the differences between English and Chinese passwords. They found that the top 5 most popular Chinese passwords made up 4.14% of all Chinese passwords studied, compared to the top 5 English passwords only making up 1.69% of all English passwords studied. A major discovery was that Chinese users prefer digits in their passwords compared to English users. Pinyins and dates appeared more often in Chinese passwords as well. It was generally believed that Chinese passwords were stronger than English ones on average, but a Probabilistic Context Free Greammar (PCFG)-based password guesser was able to guess 34% more Chinese passwords when Pinyins and date rules were added to training.

A large-scale study by Ji *et al.* [51] investigated how easily passwords can be guessed by both standard and state-of-the-art password guessers. With 145 million real world passwords, this is one of the largest studies in this matter to date. They find that in general, brute force password guessing is unnervingly successful at guessing passwords. Guessers that use training in general out-perform brute-force guessers when training and testing with different parts of the same dataset (which they call intra-site training). Interestingly, intra-site training does not always have a better performance than cross-site training.

Expanding on part of this work, Ji *et al.* [50] also demonstrated correlations between passwords across different data sets and then between a user's password and personal information. They showed that data sets of the same language have the highest correlation scores with each other and that higher correlation scores (Jaccard index and cosine similarity) lead to a higher percentage of password cracks when one data set is used as a training list for the other, even between data sets that are different languages. The attack simulation methods used were Weir's PCFG guesser, Veras's semantic guesser, OMEN, and JtR-M. Another discovery was that a high number of passwords could be guessed based off of personal information (e.g. education, company, phone number, and address). While Ji *et al.* [50], like Dürmuth *et al.* [32], had limited access to personal information, the addition of it allowed them to guess 24.6% of CSDN and 33.2% of Duduniu within 1130 guesses.

Al-Sabah *et al.* [8] analyzed a meta-data rich leak from a major Middle Eastern bank. The leaked data included detailed personal information such as name, gender, email, addresses, nationality information, and recovery questions and answers. The passwords for 79,960 accounts were recovered using John the Ripper and Hashcat, but only 65,941 were analyzed demographically due to the limited size of less frequent demographic groups. Four different demographic groups were created based

on linguistic or cultural similarities. It was impossible to perfectly split people into linguistic or cultural groupings based off of this information as a single country may have a diverse set of languages and cultures within it. Despite not being a perfect split, the passwords made by people in these groupings demonstrated their own trends and biases. For example, people from the Arabic group were more likely to include their phone number in their password compared to the other groups. It was also observed that the passwords from the English group had higher guessability according to zxcvbn despite them using less personal information in their passwords. This could be due to cultural bias as zxcvbn uses an English-based dictionary to estimate guessability.

To aid in memorability while also creating what appear to be stronger passwords, many researchers have recommended that users use phrases to create mnemonic passwords. A study by Yang *et al.* [95] began investigating the security of mnemonic passwords. By evaluating mnemonic passwords separately from non-mnemonic passwords, the authors strove to get a better understanding of the actual strength of these password strategies. When participants were given instruction to create personalized sentences for their passwords, the resulting passwords appeared less frequently when compared to participants who were given no instruction on types of phrases to use. Common sentences and well known quotes were common otherwise, resulting in high frequency passwords. This leads to a recommendation of providing high-quality instructions and examples to increase the strength of passwords. Over 50% of the passwords in 4 groups of mnemonic models were able to be guessed in 20 attempts when the sentences and passwords from the other 4 groups were used for training. The other two groups were excluded from this evaluation because their generation rules are notably different from the other groups.

Kiesel *et al.* [53] evaluated the entropy and β-success rate of mnemonic passwords.

A large-scale sentence corpus was created using the ClueWeb12 web page crawl and this corpus was used to generate mnemonic passwords. Only English sentences were considered for this work. The study tested 18 unique password generation rules. As the generation rules became more complicated, the strength of the passwords created increased. It was also discovered that the complexity of the sentence used for the mnemonic password increased the strength of a password against online attacks, but had little effect against offline attacks. Password length increased strength against offline attacks but showed almost no improvement against online attacks. If an attacker knew the generation process of the mnemonic password then their success chances were drastically improved, even without knowledge of any sentences used.

Bonneau *et al.* [14] compared the anonymized Yahoo! password dataset distribution against a uniform distribution. They found that there is little variation in the guessing difficulty of passwords created by user groups they could identify. While the password distributions would be different, all of the user groups created weak password distributions. Registering a payment card with the account had no greater impact than demographic factors. The distribution of the Yahoo! dataset was also comparable to the distribution of the RockYou dataset, showing that these distribution similarities are not isolated to the Yahoo! dataset. Unfortunately, most of the techniques used require very large sample sizes to determine a dataset's guessability accurately.

All of the passwords at Carnegie Mellon University were studied by Mazurek *et al.* [63] to investigate trends and patterns over different demographics. The password policy at the university required a password to contain a minimum of 8 characters and 4 character classes. The passwords were three fold cross validated on a password guesser trained with two training folds and a set of public passwords. The public password set contained pruned password leaks to contain passwords following the

same policy as the university. A number of correlations were found between different demographics and password strength. For example, users associated with computer science created passwords 1.8 times stronger than users associated with the business school. It was also seen that stronger passwords correlated with higher rates of error during login attempts.

Mourouzis *et al.* [65] compared how passwords have changed over time. They compared 4 password datasets, MySpace, RockYou, phpBB, and Xato, to a dataset of bad passwords compiled for this work. They calculated the Levenshtein distances of passwords from each for the 4 datasets from the dataset of bad passwords and compared the distributions. They found that passwords created later tended to be less similar to the bad passwords. However, the bad passwords could have been based on older password data, which would explain why this happened. The authors argued that users are becoming more aware of security, leading to this positive change. The passwords in the datasets were then clustered by mapping the passwords to a 10-dimensional vector and performing k-means clustering analysis. This showed an improvement over time with respect to password length and mixing character types. The distributions of characters chosen were shown to be similar between all of the datasets.

With all of the vulnerabilities of passwords discussed, it is little wonder why researchers have been investigating alternative forms of authentication. These alternative forms could alleviate the issues found in passwords and generally decrease the risk of a breach to a user's account. Still each form of authentication seems to have its own issues, so it is a matter of if the benefits outweigh the issues.

2.3 Alternative Authentication

Passwords aren't the only form of authentication, they are merely the most widely used. Authentication schemes generally fall into one of three categories: something you have (e.g. software tokens), something you are (e.g. biometrics), or something you know (e.g. passwords). While passwords are the most common form of authentication currently researchers have been investigating alternate forms of authentication that may replace them.

The main principle behind the "something you have" form of authentication involves giving the user something to use to authenticate themselves as needed. This can come in many forms such as identification cards [22] or software tokens [91]. The user is not required to remember any secret like they have to with passwords, they just use their authentication item as needed.

In 2002, Chien *et al.* [23] proposed an authentication scheme using identification cards. These cards may or may not have an associated password with them (e.g. a building access card versus a debit card with a PIN), but the card itself acts as an important tool in the authentication process. The identification card is designed to have authenticating information for the user on it. Even with an associated password with the identification card, the memory strain on the user is reduced as there is only a single password associated with the card.

On the other hand, security tokens were proposed in a patent by Weiss [91]. The patent describes a device used to calculate non-predictable codes. These codes are generated by a fixed variable, which acts as an identifier for a user, and at least one dynamic variable, which acts as a random addition to affect the result (e.g. the time of day of the request for authentication). This results in periodically changing, verifiable identification codes which are unique to a given user. While these tokens

seem incredibly strong, if the fixed variable is somehow leaked then the identifiers are easily bypassed, an event famously seen in the past with SecurID [16].

The "something you are" form of authentication, commonly referred to as Biometrics, leverage physical aspects of a user to authenticate. This can include aspects such as the user's fingerprint [72], iris [28], or even face [47]. In contrast to passwords, biometrics don't require users to recall anything and instead take information from the user that the user inherently has.

Fingerprint recognition authentication often store minor details or imperfections from a fingerprint, called minutiae, removing the requirement to store the raw fingerprint image [72] as these minutiae are claimed to be unique across individuals. Despite these claims, there are some major issues with fingerprint authentication. Matsumoto *et al.* [62] discussed the impact of targeted attacks on fingerprint systems using artificial, or "gummy", fingers. They created false fingers out of gelatin, using molds of their own fingers, and discovered extremely high rates of success unlocking systems with these gelatin fingers. Furthermore, they recreated fingerprints from fingerprints left on glass surfaces with similar results. These results are troubling if a specific user is targeted by an attacker.

Ross *et al.* [72] pushed these vulnerabilities even further by demonstrating how to reconstruct a fingerprint using minutiae data. While they do not reconstruct fingerprints in their entirety, they discussed the information that would be required to do so. Further discussion emphasized how reconstructed fingerprints could be used to create synthetic fingerprints which could compromise the authentication system. Unlike passwords, when a user's fingerprint gets stolen it cannot be replaced easily to re-establish security.

Iris recognition has also been suggested as a form of authentication. An alternate work by Ma *et al.* [59] demonstrates how the texture of the iris can be examined in

order to authenticate a user. Their method is surprisingly effective when handling a variety of low quality images, such as images with motion blur or severely obstructed by eyelashes and/or eyelids. These textures are, like fingerprints, believed to be unique to a person, even between identical twins.

Work by Daugman [28] improves on this concept by focusing on the shape features of the iris, which are quickly and easily gathered from an image. Random characteristics and irregular details are shown to be better represented than by the texture features used by Ma *et al.* [59]. The accuracy of scanning an iris has been found to be relatively high and has also been similarly shown to work on low quality or unfocused images of the iris.

A user's face has been shown to also be used for authentication. Jain *et al.* [47] discussed methods for detecting and analyzing faces from images or video. Facial recognition schemes tend to follow one of two different approaches: identifying the location of prominent facial features such as eyes, nose, and mouth or an overall analysis of the face as a weighted combination of images from a facial database. However, Jain *et al.* [47] discuss how their work is currently only suitable for certain appearances as it cannot interpret new appearances, such as a different facial angle or expression, unseen in training.

However these are just a few of the many different types of biometric authentication. Jain *et al.* [48] and later Bhattacharyya *et al.* [11] discussed an overview of many different types of biometric authentication, as well as some limitations of the technology at the time. They both discuss how biometrics seem to be a perfect solution to the fundamental issues of passwords, poor accuracy could provide a false positive authentication (e.g. facial recognition between identical twins) or false negative authentication (e.g. poor quality prints from a finger cannot be matched) are major concerns. Furthermore, there are high privacy risks for users in the case of a

data breach of biometric information as biometrics cannot be changed or replaced.

While passwords are dominant in the "something you know" authentication form, other forms exist as well such as graphical passwords [94], video passwords , or geographic authentication [79]. These schemes generally require the user and the authentication system to share a secret with each other, which the user must remember to authenticate in the future.

While there are a number of different types of graphical password authentication schemes, Wiedenbeck's PassPoints [94] have received a fair amount of attention. Graphical passwords require a user to click on different points on an image, those points acting as their password. These schemes take advantage of the image in question acting as a "cue" for a password, as well as the fact that studies have shown that images are easier to recall than words [67, 75]. PassPoints itself improves on previous graphical password schemes by allowing any image and, more impactfully, altering how the image is discretized by using three grids instead of one. By using three grids, one can be assured that every possible click a user makes will be safely within the bounds of a square of at least one of the grids. However, it has been demonstrated that many images have common points that users will click on [80], drastically reducing the security of such a technique. Further work by van Oorschot *et al.* [84] introduced fully automated attacks on PassPoints demonstrating unseen vulnerabilities in the authetication.

Geographic passwords, such as Thorpe *et al.*'s GeoPass [79], use geographic location as an important aspect of authentication. GeoPass [79] requires users to choose a geographic location on a map and that chosen location becomes their password. Similar to graphical passwords, geographic passwords take advantage of cues and image recognition, except the "image" in this case is a map. While authentication errors were rare with GeoPass, it is susceptible to social engineering attacks. MacRae *et*

al. [60] expanded on the work of Thorpe *et al.* [79] and developed GeoPassNotes. This system requires both a location and an annotation with increased security compared to GeoPass without a significant loss in memorability. Addas *et al.* [6,7] implemented geographic data both to aid in the memorability of passwords in a system known as GeoHints [7] and in a fallback authentication system called GeoSQ [6].

Despite research into alternative authentication schemes, passwords remain a dominant form of authentication in the wild. It appears replacing passwords isn't quite as simple as coming up with a new form of authentication. Bonneau *et al.* [15] analyzed and created a framework for comparing different authentication schemes. With this framework, they explain major factors in why it is so difficult to replace passwords. Some main strengths of passwords include their simplicity in implementation and familiarity to users. Many of the other authentication schemes in this comparison only offer minor improvements over passwords and therefore have little hope of replacing them.

2.4 Password Policies

With passwords here to stay, some have been looking for ways to improve the strength of the passwords that users create during the time of creation. One large branch of research in this field belongs to the study of password composition policies, or simply password policies. Password policies restrict the passwords that users can create by imposing rules they must abide by for a password to be accepted [77]. A policy may include such rules as a minimum length requirement or requiring numbers and uppercase letters. The goal of these rules is to prevent users from creating the weakest of passwords.

To demonstrate the effectiveness of password composition policies, Campbell *et*

al. [18] compared how passwords created with a password policy compared against those without. The passwords created under the enforcement of a password policy were notably different from dictionary words, implying a greater resistance against a dictionary attack against such passwords. Campbell *et al.* [19] later demonstrated this increased resistance.

Despite the purpose of password policies, they have been shown to have their own issues. The works by Campbell *et al.* [18,19] each discuss how the use of password policies does not discourage the use of personal information in passwords. Furthermore, they discuss how users find the passwords created under the enforcement of password policies are considered more difficult to remember. A user study by Inglesant *et al.* [46] demonstrated poor habits that arose in response to restrictive password policies. Users expressed frustration towards complex password policies and confirmed the results from Campbell *et al.* [18,19] that the passwords created under such restrictions were far less memorable. Despite users understanding of secure password habits, users used insecure techniques in order to remember the passwords they created such as writing their password down.

There are a number of negative user behaviours when it comes to creating passwords. A study by Campbell *et al.* [20] demonstrated how even highly restrictive policies don't reduce the amount of personal information users put in their passwords and how simple coping strategies are often used to get around restrictions. Furthermore, a number of users reported that the password they created was at the very least similar to other passwords they have, to aid in memorability of their password.

Recently, a password policy by Guo *et al.* [40] was designed to help increase the memorability of passwords while also having the structure of a restrictive policy. This policy combines the restrictiveness of overly restrictive policies with the memorability of keyboard patterns and images. The policy requires at least 8 characters and char-

acters from at least 3 of the 4 character classes, which is typically considered a strong policy, but also requests the user draw a picture on their keyboard to follow for key presses. Users perceived the passwords they created with the Optiwords policy to be both comparably strong to a random password, but also comparably memorable to a simple 8-character password.

Komanduri *et al.* [56] performed a large-scale study to investigate password strength and user behaviour in the presence of different password composition policies. They discovered that while stricter policies do result in users creating stronger passwords, there was also a significant increase in negative user behaviours. Examples of these behaviours include reusing passwords, coming up with simple coping strategies (such as appending a '!'), confirming the results from Campbell *et al.* [20]. Their study however found previously non-discussed negative behaviours, such as simply giving up on creating a password. Despite all the negative results from restrictive password policies, Komanduri *et al.* [56] did demonstrate that adding a restriction blocking dictionary words from passwords noticeably increased the strength of the passwords without adding negative behaviours.

This decrease in usability is likely why many large social-media websites choose to have less restrictive policies, as observed by Florencio *et al.* [35]. These websites are in a competitive market and desire users to be able to access them easily, which means allowing for easy to remember passwords. This is a concerning trend however as large social-media websites are some of the most attacked and breached websites. In general, we sites that accepted advertising, purchase sponsored links and where the user has a choice showed strong inverse correlation with password policy strength. Websites that don't exist in competitive markets, such as government or university websites, could afford the poor usability of a restrictive password policy, and often do have them as a result.

With all of the usability issues with restrictive password policies, it is little wonder why the National Institute of Standards and Technology (NIST) has been relaxing their recommendations over the years [17, 33, 70]. The cost of restrictive policies to users is just too great, especially as the number of accounts that users manage tends to increase as time goes on. While it is still recommended to have a password policy, to prevent incredibly weak passwords such as 3 character passwords, the general recommendations are much more relaxed. However, this still leaves us with the issue of how to convince users to create stronger passwords.

2.5 Password Meters

A second large branch in the research of proactively increasing password strength is the study of password strength meters, or just password meters. Password meters attempt to proactively measure the approximate strength of the password a user inputs to inform them if they have a weak password. If users can see how weak their password is hopefully they will adapt and create stronger passwords. Furthermore, it is believed that interacting with a password meter gives users the opportunity to discover what does and doesn't make a strong password through experimentation.

It has been shown by Ur *et al.* [82] that the presence of a password meter does promote users to create stronger passwords when compared to no meter at all. Their study compared a number of different password meters as well and how password creation was affected. The presence of a password meter increased the length of passwords compared to no meter, but the differences in visualization of the meter did not cause any significant change. While passwords created using a more stringent meter were shown to be both longer and contained more character classes, users also expressed more frustration with the meter compared to the others. Furthermore,

the presence of a password meter caused users to create stronger passwords, but the most stringent password meters created the strongest passwords. The only password meter with negative results was the text feedback only password meter, implying that a visual component is important in how users understand the strength of their password.

Despite the promising results of this study, unfortunately it has been shown that most meters used in practice don't accurately reflect how resistant the password is to being guessed. To demonstrate this, de Carné de Carnavalet and Mannan [29] studied 11 prominent web service providers from a number of different categories. Different meters are shown to give inconsistent strength scores to the same password which may likely cause confusion for users if their password is given a wide variety of strength scores. Furthermore, several meters gave high strength scores to relatively common and easy to guess passwords. Even if a password is labeled as weak, in most meters it can be trivially modified (e.g. using 'L33T' modifications) to obtain the highest strength rating. Many of these meters use simple heuristics to determine the strength of passwords, which can be overcome easily if you can determine the heuristics in use.

Part of the work in the large-scale password study by Ji *et al.* [51] (discussed in Section 2.2) also investigated the accuracy of password meters. Their results compliment the results of de Carné de Carnavalet [29], demonstrating how many passwords which are classified as strong or good strength are still vulnerable to current guessing techniques. The differences between strength score and actual password strength is feared to cause a user to be overconfident in the strength of their password. They also discuss how different meters output different strength scores for passwords, which could be confusing for users. Recent developments in password meter research focus on non-heuristic approaches, though a more advanced heuristics-based password strength meter [93] has also shown much promise.

The password meter known as *zxcvbn* was first presented by Dropbox back in 2012 [92], but Wheeler [93] published an improved version later in 2016. Despite being a heuristics-based password meter, Wheeler [93] applied advanced heuristics to ensure that *zxcvbn* displayed the strength of a password more accurately than commonly used meters. The meter assumes an attacker knows the pattern of a password and attempts to approximate how many guesses it would take for an attacker to guess the password with that knowledge. He compared the results of the meter against commonly used password guessers in research at the time to demonstrate the accuracy of his results. Between these results and the relatively low resource requirements of the guesser, it is little wonder why the interest in this meter has steadily increased over the years [71].

Heuristics aren't the only explored method for estimating password strength. Standard Markov modeling techniques have been used by Castelluccia *et al.* [21] to create an adaptive password strength meter. This was done by generating n-grams for each new password added to the database, with some additional noise for security. The probability of the password being guessed is then calculated using the n-gram database and that acts as the score for the password. As more passwords are added to the database, the n-gram database generates a more accurate distribution. The noise added does reduce the accuracy, but increases the security for the passwords in the case that the n-gram database is leaked.

A password guess number calculator was introduced by Kelley *et al.* [52] which determines if and when a given password guessing algorithm, trained on a given password training set, would guess a given password. This guess number calculator had two implementations, a brute-force Markov (BFM) model and a PCFG model. The BFM model could calculate the guess number by ordering the probabilities of one character following another. The PCFG model created a lookup table (capped at 50 trillion guesses) to calculate the guess number. The lookup table was both

time and space intensive to make, but the PCFG algorithm outperformed BFM in all cases. Training data choice showed significant effect on guessing passwords for stronger password policy but showed less impact for weak password policies. Adding the Openwall dictionary to the training actually decreased the guessing speed for certain password policies. Passwords created under a certain policy had a different guess resistance compared to passwords selected from a different group that met the rules of that policy.

A large-scale study on a feedback driven password meter by Ur *et al.* [81] was performed to test the effects of the meter on password creation. Twenty-one heuristics were used to evaluate a password and provide written feedback to the user on ways to improve. The neural network from Melicher *et al.* [64] (discussed in Section 2.6) was used to simulate an attacker on the passwords for determining how far to fill the password strength bar. They found that their feedback meter greatly improved the strength of passwords created in a weaker password policy, however it was shown that there was not a significant improvement for passwords created in a stronger password policy.

Another aspect to the work by Ji *et al.* [50] (discussed in Section 2.2 was to measure the strength of passwords using personal information, called SocialShield. SocialShield was designed to assist users in creating passwords unrelated to their social profile and thus creating more secure passwords. While SocialShield was designed to be able to quantify the correlation between a password and the user's profile information it was considered unethical to crawl users' profiles online using their leaked email information so SocialShield was only tested on username and email data. Still, by comparing the strength of leaked passwords using SocialShield to simulated attacks using personal information against these passwords in their experiments, they demonstrated how the accuracy and effectiveness of measuring the strength of passwords

with SocialShield.

When created correctly, password meters provide users with both a learning tool and a guide for creating stronger passwords. This is likely why, anecdotally, I find their use becoming more and more widespread in web services over the years. They influence users to generate stronger passwords without causing as much frustration as password policies as they are suggestions rather than rules. This allows websites that exist in competitive markets to increase the security of their users without the usability issues encountered with restrictive password policies.

2.6 Password Guessers

Password policies and password meters attempt to increase the strength of the passwords that users make against attacks from a malicious attacker. The main issue with this concept is that we do not know what methods attackers use to guess passwords. The safest guess we have is that attackers use the best methods and the most resources possible in these attacks, and to keep up with this idea, researchers create their own password guessers in an attempt to stay at least on par with attackers. If malicious attackers end up using researcher's technology then at least we should have an accurate understanding of their capabilities.

Using Markov models to guess passwords was first implemented by Narayan *et al.* [66] back in 2005. Markov modeling techniques from Natural Language Processing were used to reduce the size of the password space to be searched. This took advantage of the fact that the distribution of letters and adjacent letters in weak passwords are likely to be similar to the distribution of letters in the the users' native language. This attack used regular dictionaries as well as rules for common patterns and modifications of passwords to increase their coverage of the plausible password space. This method

cracked 67.6% of the passwords from Passware, a vast improvement over the Rainbow attack's 27.5%, which was the fastest password guessing technique at the time.

A number of variants to the standard Markov model technique for password cracking were later introduced by Ma *et al.* [58]. These variants include differing by order for Markov chains and approaches to smoothing among other distinctions. These variations were all tested in different training/testing scenarios where different combinations of datasets were used for training and testing. The datasets used were RockYou, Yahoo, PhpBB, Duduniu, 178, and CSDN. Different variants performed better in the different situations. The backoff model with end-symbol normalization performed the best overall among all scenarios. For most models, distribution-based normalization outperformed all other normalizations, except in the case of the backoff model. For English datasets, order-5 Markov chain model performed the best but with Chinese datasets order-3 and order-4 often outperform order-5.

Dürmuth *et al.* [31] proposed further improvements to the standard Markov modelling technique. The main contribution to their technique, called the Ordered Markov Enumerator (OMEN), is the fact it outputs passwords in decreasing order of probability. By making this change, the most likely passwords will be selected first for guessing. Their algorithm is adaptive, keeping track of the success rates of different password lengths and favoring the password lengths that are more frequent. OMEN has significantly improved guessing speed and accuracy when compared to John the Ripper and a PCFG password guesser.

Weir *et al.* [90] first described a method of guessing passwords using PCFGs. A training set of passwords is first decomposed into structures. These structures break the passwords down into representations of sequences of characters separated by character type. The three character types they use are letters, digits, and symbols which are represented by L, D, and S respectively. These structures are ordered in

decreasing probability of their occurrence, then used for password guessing. The method managed to crack between 28% to 129% more passwords on the MySpace dataset when compared to John the Ripper. This technique is also one that many future password guessing techniques compare themselves against and acts as a basis for other guessers.

Veras *et al.* [85] implemented Natural Language Processing techniques in order to create a framework for analysis of the semantic patterns in passwords. This knowledge is then used to generate password guesses in offline attack scenarios more efficiently than other approaches at the time by building upon previous work in PCFGs. This method has a longer run time than expected which requires further study to detect where this issue originates from. The guessing technique also has increased memory consumption compared to other approaches. Despite these losses, the semantic approach managed to guess over 67% more passwords from the LinkedIn leak and 32% more passwords from the MySpace leak when compared to both PCFGs and John the Ripper.

More recently, recurrent neural networks have been used by Melicher *et al.* [64] in order to guess passwords which work as well or better than many current approaches to password guessing. While the neural network can be used as a standard password guesser, a main strength of it is in password strength estimation. The neural network uses Monte Carlo simulations to quickly and relatively accurately calculate the guess number of a password, which Ur *et al.* [81] implemented in their password meter (described in Section 2.5). They demonstrated the strength of the neural network as a password guessed by comparing it against typically studied password guessers (e.g. PCFGs, John the Ripper). The neural network outperformed all other guessers after around 10^{10} guesses and matched the other methods before that point.

Deep learning has been used by Hitaj *et al.* [43] to create a Generative Adver-

sarial Network (GAN) called PassGAN. This technique differed from the Recurrent Neural Network created by Melicher *et al.* [64]. Each guesser in their experiments was tuned, if possible, to guess the passwords of the testing datasets as effectively as possible, using prior knowledge of the datasets. PassGAN however does not use prior knowledge or tuning to generate optimal passwords. Despite this lack of tuning, PassGAN performed comparably to the state-of-the-art guessers in their experiments. Furthermore, PassGAN was shown to be able to guess an incredibly high number of guesses compared to other guessers they studied (e.g. Hashcat, Weir's PCFG).

Ur *et al.* [83] compared the best practices at the time between John the Ripper (JtR), Hashcat, PCFGs, Markov models, and a professional password cracker from KoreLogic. This research differed from earlier findings that often used stock configurations for comparison between guessers. Markov and PCFG approaches often outperformed JtR and Hashcat early on. Hashcat and JtR often showed rapid improvements in guessing after 10^{10} guesses. Due to large resource requirements, the Markov technique could not make more than 10^9 guesses. The professional allowed their method to run for 10^{13} guesses before adding freestyle rules based on results. This produced a large increase in guessing effectiveness unseen in all of the automated techniques and in the end were more successful than any of the automated guessers.

2.7 Literature Gap

We note that system administrators need to make many decisions to implement effective password checking. These decisions include which subset of guessing tools to choose among many available options, how to train them, which training dataset to choose, etc. To support these decisions, the literature falls short in systematically understanding guesser behaviours and their ability to compliment or substitute one

another. This work attempts to address this gap. We focus on developing metrics and statistics to aid understanding password guessers (both present and future), to facilitate the work and decision making of administrators for password checking.

With begin with inspiration from the work of Ji *et al.* [50] in calculating the similarity between training and testing datasets. We decided to take their work further and create a framework to investigate how and why password guessers create the guesses they do. Calculating the similarity between modified features or exact passwords in a variety of situations we can begin an in-depth investigation of how password guessers are the same, and how they might be different. From this point, instead of creating our own traditional password guesser we demonstrate how to these metrics can be used to identify guessers that show promise when used together. With these results, we find low-resource password guessers that demonstrate comparable results to state-of-the art guessers used in research today. This work is designed to be applicable to any past, present and future password guesser.

Chapter 3

Metrics and Algorithms

Our primary objective is to gain a deeper understanding of password guesser behaviour by comparing the actual password guesses produced by each guesser across pairs of real-world testing and training password datasets. To accomplish this we must devise methods of objectively comparing and analyzing this behaviour. We begin by defining common notation used in our metrics. Following this, we describe how we extract structural features from passwords for use in structural analysis and describe the statistics we use to compare two password lists. Finally, we describe the different statistics and how we use them in our analysis.

We consider a set of m password guessers $\mathcal{G} = \{g_1, \ldots, g_m\}$ where each g_i represents a specific guesser (e.g., John the Ripper, OMEN, etc.). By ensuring $\mathcal{G}$ includes diverse and powerful set of guessers, we aim to understand how each guesser behaves when trained on or tested against particular password datasets, what types of passwords they guess, and how similar one guesser's behaviour is to others. To this end, each guesser $g \in \mathcal{G}$ will be trained on and tested against a set of n password datasets $\mathcal{D} = \{D_1, ..., D_n\}$, where each D_j is a publicly-available password dataset (e.g., Rock-

You, Twitter, etc.).[1] When a guesser $g_i \in \mathcal{G}$ is trained on a dataset $D_j \in \mathcal{D}$, it can create a list of password guesses L_{ij}. To compare various guessers trained on various datasets, we extract (structural) features from L_{ij}, and develop some metrics and statistics. To achieve this, we first define our features on each password, then aggregate them for the list of passwords, and then compare the aggregated features.

3.1 Password Features

For each password w, we extract two structural features: *password length* n_w (i.e., the number of its characters) and the *number of character classes* c_w that it contains. We focus on four distinct character classes: lowercase letters, uppercase letters, numbers, and symbols. For instance, $w = passw0rd!$ has $n_w = 9$ and $c_w = 3$ with three character classes of lowercase letter, number, and symbol.

To extract features from password list L' (e.g., leaked password database or guess list of a guesser), we first aggregate the extracted features of all $w \in L'$ into a matrix $\mathbf{V} = [v_{xy}]$ where v_{xy} is the fraction of passwords in password list L' which contains y characters covering x character classes:

$$v_{xy} = \frac{1}{|L'|} \sum_{w \in L'} \mathbb{1}[c_w = x \ \ \& \ \ n_w = y], \tag{3.1}$$

where $\mathbb{1}[.]$ is the indicator function, and $|L'|$ represents the number of passwords in the list.[2] The matrix $\mathbf{V}$ has a natural probability interpretation: when one selects a password w from the password list L' uniformly at random, the password w contains y characters from x character classes with a probability of v_{xy}. In other words, our matrix $\mathbf{V}$ captures the joint probability distribution of passwords over character

[1] We use the terminology of "testing against a dataset" when a guesser is guessing the passwords of a target password dataset.

[2] Indicator function $\mathbb{1}[s]$ returns 1 if the statement s is true; otherwise 0.

classes and the number of characters.

For example, let's assume we have a password list L=['abc', 'abc1', '1qw', 'pass', 'abc', '1234', 'qwe']. These passwords fall into the following categories:

$c_w = 1$ & $n_w = 3$	$c_w = 1$ & $n_w = 4$	$c_w = 2$ & $n_w = 3$	$c_w = 2$ & $n_w = 4$
abc	pass	1qw	abc1
abc	1234		
qwe			

which results in the following matrix V:

$$V = \begin{bmatrix} 0 & 0 & 3/7 & 2/7 \\ 0 & 0 & 1/7 & 1/7 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

We converted all of our password lists into 4 by 50 matrices by grouping all passwords with a length of 50 or more into a 50+ length group, placing each password in one combination between 4 character class categories and 50 length categories. To ease our notations and analyses, we collapse (i.e., flatten) the matrix $\mathbf{V}$ into a 200-dimensional feature vector $\mathbf{v}$. We refer to this feature vector as the *structural features* of a password list. This simple representation allow us to preserve the impact of password policies of each password list.

3.2 Statistics

Our metrics for comparing two password lists (e.g., leaked datasets or guess lists) use either the structural features (discussed above) or the passwords shared between two

lists. Our deployed metrics have been widely used in information retrieval [9, 37, 76], data mining [10, 30], and other password research [50].

3.2.1 Cosine Similarity

Cosine similarity measures the angle between two non-zero vectors. For comparison of two password lists, one can extract structural features from each list, and then use the cosine similarity on the corresponding feature vectors. The cosine similarity between two password lists A and B is given by

$$C(A, B) = \frac{\mathbf{v}_A \cdot \mathbf{v}_B}{\|\mathbf{v}_A\|\|\mathbf{v}_B\|}, \tag{3.2}$$

where $\mathbf{v}_A$ and $\mathbf{v}_B$ are structural feature vectors of A and B, respectively. $\|.\|$ is the Euclidean norm, and $\mathbf{v}_A \cdot \mathbf{v}_B$ is the dot product of those two vectors. The closer the cosine similarity value is to 1, the smaller the angle between the two vectors is, and the more similar they are. In other words, two lists of passwords with similar feature distributions have a high cosine similarity. As with Ji *et al.* [50], we apply our methodology to compare pairs of password datasets. By doing so, in our experiments, we show how various leaked password datasets compare to each other in their structural characteristics. While Ji *et al.* also applied cosine similarity to their feature vectors, our chosen features differ from theirs. We chose to focus on character classes and password length as the basis for our features to allow for a straight-forward comparison with a system's password policy. One can easily see the presence of a password policy (e.g., 12 characters, 1 digit, 1 symbol) in our vectors, and readily compare them to those of different datasets or guess lists. In practice, this would allow a system administrator to immediately compare a prospective training dataset with their own system's policy in a more clean fashion than is possible with other features (e.g.,

n-grams, grammar structures, dictionaries).

3.2.2 Jaccard Index

Jaccard index measures the extent two sets overlap with each other, where the intersection of two sets is compared to their union. The Jaccard index between two password lists A and B can be computed by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3.3}$$

The closer the Jaccard index is to 1, the closer in size the intersection of the sets is to their union, and consequently the more similar two sets are. In other words, the two sets of passwords with high amounts of overlap will have a high Jaccard index. The Jaccard index also has a natural probabilistic interpretation: if one chooses a password uniformly at random from either password lists, the Jaccard index captures the likelihood of selecting a password belonging to both sets. One can utilize this same methodology for comparing two password datasets. In our experiments, we use Jaccard similarity to investigate how passwords in various password datasets compare to each other (or specifically, overlap with each other).

3.2.3 Generalized Jaccard Index

When password lists have duplicates (e.g., leaked password datasets), we view the password list as a multiset, a modification of sets that allows for duplicated elements. In these cases, we apply a generalized version of the Jaccard index [50] to preserve the frequency information of password duplicates in password lists (e.g., comparing raw leaked datasets) which works on vectors which take into account both the frequency of repetition and overlap instead of on sets. In this form the index compares two equally

sized vectors composed of the number of occurrences of each password found in the union of passwords between the two sets. This version allows for a more accurate comparison of datasets with duplicates when compared with converting the datasets to sets and using the canonical Jaccard Index. Letting $f(w, A)$ be the number of occurrences of password w in password list A, the *generalized Jaccard index* between two password lists A and B is given by

$$J(A, B) = \frac{\sum_{w \in U} min\left(f(w, A), f(w, B)\right)}{\sum_{w \in U} max\left(f(w, A), f(w, B)\right)}, \tag{3.4}$$

where $U = A \cup B$.

As both the cosine similarity and the Jaccard index are symmetric, they are computed once for each pair of password lists.

Our comparison metrics can be readily used for the comparison of a pair of password lists generated by two guessers. However, to compare two guessers thoroughly, one requires some statistics to summarize the comparison metrics of two guessers under many different settings (e.g., under different training and testing datasets). This section explains our proposed statistics for summarizing comparison metrics. Our statistics fall into two categories.

3.3 Guessing Success Statistics

The *guessing success* statistics quantify the guessing accuracy of guessers under various settings (e.g., training and testing datasets), and also determine how training data affects guessing success for various guessers.

3.3.1 Success Rate

When each guesser g_i is trained on password dataset D_j and tested against password dataset D_k, one can compute its *success rate*, as the portion of successfully guessed passwords, by

$$s_{ijk} = \frac{|L_{ij} \cap D_k|}{|D_k|}. \tag{3.5}$$

where L_{ij} is the password guess list created by guesser g_i when trained on password dataset D_j. Note that $s_{ijk} \in [0, 1]$, where $s_{ijk} = 1$ implies that all passwords in D_k are guessed successfully by g_i trained on D_j. This is the commonly used comparison between guessers in literature. From this point, we branched out to understand different ways success rate is affected by different circumstances.

3.3.2 Guesser Success Rate

To summarize the success rate for a specific guesser g_i, one can compute its mean success rate over all distinct training and testing datasets by

$$\langle s_{i::} \rangle = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k=1;k\neq j}^{n} s_{ijk}. \tag{3.6}$$

We also calculate the standard deviation of the success rate for a specific guesser g_i over all distinct training and testing datasets by

$$\langle sdev_{i::} \rangle = \sqrt{\frac{1}{n(n-1)-1} \sum_{j=1}^{N} \sum_{k=1;k\neq j}^{N} (s_{ijk} - \langle s_{i::} \rangle)^2} \tag{3.7}$$

3.3.3 Training Success Rate

We similarly compute the success rate of training dataset D_j by altering the equation to

$$\langle s_{:j:} \rangle = \frac{1}{m(n-1)} \sum_{i=1}^{m} \sum_{k \neq j}^{n} s_{ijk}. \tag{3.8}$$

3.3.4 Fixed Success Rate

Finally, one can compute the the average success rate for a fixed dataset D_j and guesser g_i by

$$\langle s_{ij:} \rangle = \frac{1}{n-1} \sum_{k \neq j}^{n} s_{ijk} \tag{3.9}$$

3.4 Guessing Behaviour Statistics

This class of statistics are devised to either compare the guessing behaviours of password guessers with each other, or measure how different training datasets affect the guessing behaviour of a given guesser.

3.4.1 Guessing Similarity

Our *guessing similarity* statistic summarizes the similarity of two guessers' guess lists when trained on the same dataset by averaging the comparison metric (e.g, Jaccard or Cosine) of their guess lists over various training datasets. We calculate the *guessing similarity* of two guessers g_i and g_j by

$$G(g_i, g_j, M) = \frac{1}{n} \sum_{k=1}^{n} M(L_{ik}, L_{jk}) \tag{3.10}$$

where $M \in \{C, J\}$ is either Cosine similarity (see Eq. 3.2) or Jaccard index (see Eq. 3.3), and L_{ik} is the list of password guesses (without any duplicates) generated by

g_i trained on datasets D_k. Here, n is the number of datasets in $\mathcal{D}$. We also introduce *Successful guessing similarity* to measure how two guessers' successful guesses are similar:

$$SG(g_i, g_j, M) = \frac{1}{n(n-1)} \sum_{k=1}^{n} \sum_{\ell \neq k}^{n} M\left(L_{ik} \cap D_\ell, L_{jk} \cap D_\ell\right). \tag{3.11}$$

3.4.2 Training Similarity

One might be interested in measuring how similarly two different password datasets can train guessers. To this end, we introduce our *training similarity* statistic which calculates the extent two different training password datasets result in generating similar guess lists of passwords when used for training. We define *training similarity* between two datasets D_j and D_k by

$$T(D_j, D_k, M) = \frac{1}{m} \sum_{i=1}^{m} M(L_{ij}, L_{ik}), \tag{3.12}$$

where m is the number of different guessers in $\mathcal{G}$. This formula computes how similarly D_j and D_k can train guessers on average. By capturing the extent two various datasets are effectively similar in training guessers, one can identify training datasets which are as effective as another dataset in training guessers. This could be used to identify effective, yet small datasets, which could drastically speed up the training process.

3.4.3 Training Independence

To investigate how sensitive each guesser's output is to the choice of training data, we introduce the *training independence* statistic. This statistic is computed by averaging the comparison metrics (Jaccard or Cosine) between of all pairs of password guess lists created by the same guesser but with different training datasets. We formally define the *training independence* of a guesser g_i by

$$I(g_i, M) = \frac{1}{n(n-1)} \sum_{j}^{n} \sum_{k \neq j}^{n} M(L_{ij}, L_{ik}), \tag{3.13}$$

where n is the number of datasets in $\mathcal{D}$, and $M \in \{J, C\}$. This formula computes, on average, how similar two password guess lists generated by guesser g_i are, when trained on two different password datasets. As this value approaches 1, password guess lists become more similar, regardless of the choice of training data. A password guesser with a training independence value of 1 doesn't learn from the training data as it always generates the same guesses despite the choice of training dataset (e.g., a brute force attack). A guesser with a very low training independence value would not generalize effectively beyond the training data (e.g., JtR-Wordlist mode with no rules). It should be noted that a training independence of 0 would be possible for JtR-Wordlist mode with no rules if the training datasets are disjoint.

Chapter 4

Experimental Methodology

Our experiments are designed to enhance our understanding of different password guesser's behaviour when we vary the training and testing datasets and when each tool is used to complete different password guessing tasks. In particular, we aim to understand how each guesser performs when attacking different password datasets with varied training data (e.g., varied cross dataset similarity and training dataset size), what types of passwords they generate, how similar their guesses are to each other, and how this changes in online and offline attack scenarios. To this end, we choose a variety of different password guessers and password datasets.

4.1 Comparison Resources

We design our experiments to enhance our understanding of different password guessers' behaviour when we vary the training and testing datasets. To this end, we first choose a variety of different password guessers and password datasets.

	Number of Passwords			
Datasets	**Total**	**Unique**	**Ratio**	**Type**
ClixSense [39]	2,222,359	1,628,205	0.7326	Plaintext
Webhost [36]	15,292,021	10,589,775	0.6925	Plaintext
Mate1 [73]	27,403,932	11,988,154	0.4375	Plaintext
RockYou [24]	32,596,319	14,337,716	0.4399	Plaintext
Fling [26]	40,769,652	16,810,091	0.4123	Plaintext
Twitter [27]	40,872,901	22,579,065	0.5524	Plaintext
Merged*	159,157,184	77,933,006	0.4897	Plaintext
LinkedIn [41]	174,243,105	61,829,207	0.3548	Hashed

Table 4.1: The password datasets, their sizes, and the ratio between unique and total number of passwords. *Merged contains all other plaintext datasets in this table.

4.1.1 Password Datasets

Our experiments use a variety of publicly available leaked password datasets, which have been the subject of other password research studies (for example, [25, 38, 86, 87, 89, 96]). We have curated and cleaned these datasets by converting their passwords to Unicode. Table 4.1 shows the number of total and unique passwords in each dataset as well as the ratio between those values.[1]

Fling. This password dataset is from an adult dating website Fling, which suffered from a data breach in 2011 [26]. For most of the time before the leak, Fling required passwords to have at least one digit (though all digit passwords are allowed).

ClixSense. The online survey website ClixSense was attacked in 2016 [39]. It is our smallest dataset with only 2.2 million passwords. By training on this small password set, we can begin to see how guessers trained on small datasets perform when tested against larger testing sets.

[1]We exclusively use publicly available datasets and don't report any specific password information. Thus, there is no risk of exposing private user information. We keep only the passwords with no links to their original owner.

Twitter. This dataset, created in 2016 from credentials harvested from users of the social media website Twitter, is our largest individual dataset with over 40 million passwords [27].

Webhost. The web hosting site 000webhost suffered a breach in 2015 [36]. This dataset, with 15 million passwords, has a more complicated password policy than other password datasets. Also, its users were generally more tech savvy (e.g., system administrators) than average users of social media or dating websites.

Mate1. This dataset is collected from the 2016 breach of the dating website Mate1 [73]. As a result of the breach, 27 million users passwords were leaked.

RockYou. The defunct social media application developer RockYou suffered a data breach in 2009. RockYou's password policy only required a length of 5 characters [24]. This large dataset is widely studied in previous work (e.g., [13, 49, 55]).

Merged. We create a merged dataset of all of the previous plaintext datasets. This dataset contains more than 159 million passwords.

LinkedIn. The business-oriented social network website LinkedIn suffered from a breach in 2012 [41]. This dataset contains 174 million unsalted passwords hashed with SHA1. We use LinkedIn exclusively as a test dataset when simulating offline attacks.

4.1.2 Password Guessing Tools

To understand how different password guessers behave, we draw our attention to three different general classes of password guessers commonly-used in the literature: Markov models, Probabilistic Context Free Grammars (PCFGs), and Neural Networks. From

these classes, we select six different password guessers for our studies.

John the Ripper Markov Mode (JtR-Markov). We use its community build (1.9.0-bleeding-jumbo) [1] in Markov mode. When a guess is generated in Markov mode, its probability of being guessed is also calculated. To set our cutoff number of guesses, we set the minimum probability for passwords to be guessed through the Markov level setting. This allowed us to generate our desired number of passwords.[2] We also restrict the maximum length of passwords to 12 characters, which we determined to provide the best results and is consistent with other studies [85].[3] We pipe these guesses to a file, which we call a *guess list* for later evaluation where we use this file for a wordlist attack. This pipeline strategy is used for all other guessers discussed below. JtR runs single-threaded during both training and guessing.

Ordered Markov Enumerator (OMEN). We use OMEN [3,31] with the default settings. OMEN produces only ASCII passwords and runs single-threaded during training and guessing. OMEN [3, 31] applies a Markov approach to password generation and outputs guesses in probability order to improve guessing speed. OMEN uses training data to estimate probabilities for different n-grams and lengths of passwords. We use the default 72 character alphabet and n-grams of size 4 while applying additive smoothing. Instead of determining a probability cutoff, there is built-in functionality to choose a guess cutoff value, which we set to match the cutoff of other guessers. OMEN produces only ASCII passwords and runs single-threaded during training and guessing.

Probabilistic Context-Free Grammar (PCFGv4). We used PCFG version 4.0 [4], which is an extension of the original Weir et al. PCFG [90] that includes enhancements

[2]We use the command ./genmkvpwd *statfile* 0 12 to determine the Markov level required to generate the number of guesses we need.

[3]We use the command ./john –markov=Number:0:0:12 –stdout where Number is replaced with the Markov level determined by the genmkvpwd command.

proposed by Houshmand *et al.* [44]. A notable enhancement is its use of OMEN to generate a certain percentage of passwords and generate the remainder with PCFGs. In our experiments, we disable this feature and generate passwords exclusively from PCFGv4 (i.e., learnt grammars) as the use of OMEN decreased the success rate of the guesser significantly in online attacks at 1 million guesses and caused a minor decrease in offline attacks.[4] PCFGv4 runs single-threaded for training and guessing.

Semantic Guesser (Sem). Sem [85] is a PCFG-based password guessing tool that incorporates part-of-speech tags and word semantics in the password grammars. We use the lite and database-free version of Sem [5,85]. The grammars are trained using maximum likelihood estimation, the backoff algorithm is used for producing tags, and mangling rules are enabled for generating guesses. Similar to OMEN, this guesser has built-in functionality for choosing a guess cutoff value. The *deadbeat* algorithm [88] is used for generating guesses and the *mangle* option is used. Sem uses multiprocessing with all available CPUs during training, but runs single threaded for guessing.

Neural Network (NN). We generate guesses using "human" mode of NN [2,64]. The generated passwords are sorted in a descending order of their probabilities. We limit the length of passwords to be between six and forty characters with "basic" policy enforcement. Like OMEN, the NN filters non-ascii passwords during training and it produces only ASCII passwords. Because of our large datasets, we use a larger model than the original paper, consisting of three LSTM layers (with 1024 neurons each) and two dense layers (with 512 neurons each), which outperformed the original model in our preliminary tests. We use a context length of 10, a train/test ratio of 80:20, and backwards training option. We train each model with Adam [54] for five generations. The neural network is our only guesser that uses GPU resources along

[4]The command "python3 trainer.py -r *name* -t *training_data* -e utf-8" was used for training where the *name* is the file name for learnt parameters (i.e., grammars) and *training_data* is the location of the password list to train on.

with CPU. The neural network runs multi-threaded during training and guessing.

Identity Guesser (ID). This guesser takes a training dataset as input, removes its duplicates, and outputs its unique passwords in the descending order of their frequency in the training dataset. In other words, this guesser computes the empirical probability distribution of the passwords in the training dataset (i.e., training phase), then outputs the passwords from the highest to the lowest probability (i.e., generation phase). This simple guesser is a valuable benchmark for understanding how well other guessers learn and generalize.

For the purposes of guessing passwords we also use one more guesser:

John the Ripper Wordlist Mode. The community build (1.9.0-bleeding-jumbo) [1] of John the Ripper also has a wordlist mode. In this mode one can provide two inputs, a password guess list (as either a file or stdin) and a list of passwords to guess, with each entry in each of the lists separated by new lines. John the Ripper will try each guess from the guess list, in order, against the passwords in the list to guess. If the list of passwords to guess is hashed and the hash is known, it can also be provided as input. If no hash is specified then John the Ripper will attempt to detect the hash (if any) automatically. John the Ripper outputs both statistics on guessing (number of successful guesses, duration of guessing, etc.) as well as the list of all passwords which were successfully guessed. We use this password guesser exclusively for guessing passwords, and do not compare it's behaviours to the other guessers.

4.2 Guessing Passwords

In order to compare the behaviours of password guessers, we were required to not only guess passwords but also gather information such as what the guesses were and which guesses were successful. To achieve this, we follow a simple procedure for each

combination of guesser, training data, and testing data as follows:

1. Train the guesser on the chosen training dataset.
2. Save the output guess list of the guesser to a file.
3. Input the guess list and testing data to John the Ripper in wordlist mode.
4. Save the results of the guessing attack and the successful guesses to another set of files.

With all of this information gathered, we are able to compare guessers in the following ways.

4.3 Resource Measurements of Password Guessers

To help understand the resource requirements of each guesser, we analyze each guesser's runtime during training and guess list creation. Each guesser is trained and generates guesses on the same GPU-accelerated server which ran no other jobs. The server has 2 Intel(R) Xeon(R) Gold 6148 CPUs with 80 total cores @ 2.40GHz and 4 Nvidia GeForce 1080 Ti GPUs. Each guesser utilizes available resources to different degrees based on their approach and implementation. For training, we created two datasets by sampling 1 million and 50 million passwords from the Merged dataset. For testing, we have each guesser generate as many passwords as it can, up to 300 million guesses.

4.4 Password Guessing Success

To better understand each guesser on an individual level, we perform a series of experiments that investigates the behaviour of each guesser (i.e., generated list of guessed passwords) under various settings (e.g., trained with different datasets or

with different guess cutoffs). We examine the average success rate of each guesser across varied training data, testing data, and password guessing scenarios (e.g., online and offline attacks). We also explore how similar guessers' guess lists are to each other. We perform this analysis by applying our metrics and statistics to the guess lists generated by each guesser under varied settings.

To gauge the average performance of each guesser, we train and test every guesser on each possible pair of unique training and testing plaintext datasets. We omit the merged set as it is a composite of other plaintext datasets. We also exclude LinkedIn as it is hashed and therefore can not be trained on or analyzed in our previous dataset similarity comparisons. Then, each guesser generates guess lists at cutoffs of 1 million and 300 million guesses to simulate online and limited offline attacks, respectively, and calculate the guesser success rate (see Eq. 3.6) of each guesser.

We investigate how guessing success rates are impacted by different aspects of training datasets such as training dataset size, and the similarity between training and testing datasets. We also investigate how similarities between password datasets are preserved in their corresponding password guess lists. To accomplish this, we train all six password guessers on each of the six individual plaintext datasets and test them against every other plaintext dataset, yielding 180 password cracking scenarios. For all guessers, we set the cutoff to 300 million guesses.

We ask whether the success rate of a guesser, on average, increases with the size of training dataset. To this end we compare the *training success* (see Eq. 3.8) of each set of training data to the size of the training data itself. For a formal analysis, we further calculate the statistical correlation between the number of passwords in the training dataset and the averaged success rate.

We next focus on how the similarity between training and target datasets impacts the success rate of guessers. We first compute the cosine similarity and generalized

Jaccard index (see Eq. 3.2 and Eq. 3.4) between password datasets, and then explore the relationship of these similarities with success rates. We further calculate the statistical correlation between the similarity values and success rates. This is our only experiment that can be compared with Ji et al.'s [50], in the sense their work computes similarity using cosine similarity and Jaccard index; however, our features for cosine similarity differ, and our use of Jaccard index measures overlap of the datasets rather than overlap of features. Additionally, we use a different set of datasets and guessers.

We explore how similarly two datasets can train a guesser using our notion of training similarity (see Eq. 3.12). We exclude the Identity guesser due to its simplicity in learning; also, its results mirror dataset similarity (described above).

4.5 Password Guessing Behaviour

We investigate the behaviour of each guesser (i.e., their generated guess lists) under various training and target datasets. We also explore how each guesser compliments and substitutes others.

We use our training independence metric (Eq. 3.13) to study guessers' sensitivity to changes in the training dataset. Guessers with high training independence will produce similar guesses when trained on different datasets. The two extremes of training independence are a brute-force guesser with the maximum value of 1 and the identity guesser with the minimum value (not quite 0 due to overlap between training datasets).

One important characteristic of guessers is how well they can generalize, i.e., predict and generate previously unseen passwords. To measure this, we train each guesser on the Webhost dataset as it is the least similar to the other datasets, both in terms of structure and actual password overlap (discussed in 5.3.2). We then test

the Webhost trained guessers against every other dataset and calculate each guesser's mean success rate.

We intend to learn how each guesser's success rate is impacted by the size of training data, drawn from the same distribution. Sampling from the Twitter dataset, we create three different datasets of sizes 1 million, 10 million and 30 million. After training each guesser on each dataset, we generate guess lists at a cutoff of 300M and test them against all other datasets and report the Fixed Success Rate (see Eq. 3.9).

Using our notion of guessing similarity (see Eq. 3.10), we analyze how similar the guess lists of two guessers are when they are trained on the same training data. We investigate the cosine and Jaccard guessing similarity between guessers at cutoffs of 1 million and 300 million guesses.

One might be interested in measuring the uniqueness of *successful* guesses between guessers. To achieve this, we use our successful guessing similarity (see Eq. 3.11) with generalized Jaccard index The generalized Jaccard allows us to weight the successful guesses of each guesser based on their frequencies in the target dataset.

4.6 Combining Guessers

We evaluate the ability of password guessers to compliment one another on a previously unseen dataset (i.e., LinkedIn) in an offline attack scenario. We begin by evaluating each individual guesser against the LinkedIn dataset. Next we analyze different combinations of guessers.

To compare guessers' performance, we train each guesser on the Merged dataset, and allow them to each make 2 billion guesses against the LinkedIn dataset. From here we calculate the mean success rate of each guesser and apply the findings to improve the results using combinations of guessers.

From here, we use our previous analysis to design a *combination attack* where the Identity guesser is used to attack a password dataset prior to the application of a set of other guessers. This hybrid approach exists in John the Ripper where a traditional John attack follows wordlist mode. We run many independent combination attacks on LinkedIn, where the Identity guesser precedes a subset of other guessers. Each guesser is trained on the Merged Dataset and produce two billion guesses. For each combination attack, we begin with a wordlist attack on LinkedIn using the Identity guesser's guess list followed by the other guessers' guess lists.

Chapter 5

Experiments

To better understand each guesser on an individual level, we perform a series of experiments that investigates the behaviour of each guesser (i.e., generated list of guessed passwords) under various settings (e.g., trained with different datasets or with different guess cutoffs). We compare practical aspects of each guesser including relative runtime and resource requirements. We also examine the average success rate of each guesser across varied training data, testing data, and password guessing scenarios (e.g., online and offline attacks). We also explore how similar guessers' guess lists are to each other. We perform this analysis by applying our metrics and statistics to the guess lists generated by each guesser under varied settings.

5.1 Resource Measurements of Password Guessers

Table 5.1 reports guesser training and generation time.[1] For each guesser, the training time increases with the training dataset size. Markov-based and Identity guessers perform the fastest ($<$ 25 seconds for 50 million), with PCFGs taking longer (about

[1]Our code for training the identity guesser (i.e., computing empirical distribution of unique passwords) and its guess generation (i.e., sorting passwords based on their probabilities) is written in Python without any optimization

	Training		
Guesser	**1 Million**	**50 Million**	**Generation**
JtR-Markov	00h 00m 00.1s	00d 00h 00m 02.2s	00h 00m 33s
Identity	00h 00m 00.3s	00d 00h 00m 24.9s	00h 00m 18s
OMEN	00h 00m 03.0s	00d 00h 00m 23.0s	00h 07m 10s
Sem	00h 01m 38.3s	00d 00h 20m 14.6s	00h 55m 30s
PCFGv4	00h 03m 49.5s	00d 01h 03m 38.4s	00h 30m 58s
NN	01h 18m 08.0s	02d 17h 01m 49.0s	19h 44m 20s

Table 5.1: Guesser training and generation time. Each dataset is of different size and randomly sampled from the Merged Dataset. Guessers have generated 300M guesses.

one hour for 50 million) and the neural network taking the longest (more than two and half days for 50 million). For password generation, we observe that Identity guesser and Markov models are again by far the fastest. It should be noted that the Identity guesser only produced approximately 67M guesses, almost 4.5 times fewer guesses than produced by other guessers. The NN is considerably slower than others: 2100 times slower than JTR-Markov, and even 39 times slower than PCFGv4. We also note that on top of the high time requirements of NN, it is also our only guesser which requires GPUs as every other guesser operates using only the CPU.

5.2 Password Guessing Performance

To evaluate the performance of each guesser, we compute the average success rate of each guesser across varied training data, target data, and password guessing scenarios (i.e., online and offline attacks).

5.2.1 Impact of Password Guesser Choice

To gauge the average performance of each guesser, we train and test every guesser on each possible pair of unique training and testing plaintext datasets. We omit the

Guesser	Success Rate@1M	Success Rate@300M
Identity	23.238 (11.859)	30.519 (14.079)
JtR-Markov	0.665 (0.993)	27.591 (11.563)
OMEN	5.921 (3.225)	22.121 (10.749)
Sem	18.219 (10.344)	41.343 (13.274)
PCFGv4	23.551 (11.545)	47.397 (12.364)
NN	17.662 (11.585)	40.768 (19.734)

Table 5.2: Guessers' mean success rates at 1 Million and 300 Million guesses (standard deviations in parenthesis). The two best and worst are highlighted with green and red, respectively.

Merged set as it is a composite of other plaintext datasets, and we exclude LinkedIn as it is hashed and therefore can not be trained on or analyzed in our previous dataset similarity comparisons. Then, each guesser generates guess lists at cutoffs of 1 million and 300 million guesses to simulate online and limited offline attacks, respectively. Table 5.2 shows the mean success rate and standard deviation of each guesser, computed by Eq. 3.6 and Eq. 3.7 respectively. At one million guesses, PCFGv4 and Identity outperform others, while JtR-Markov and OMEN perform the worst. Notably, only PCFGv4 is able to outperform Identity at this cutoff with a negligible margin. This suggests that, for online attacks, only PCFGv4 provides a slight improvement over using the training data as a wordlist. Both Markov-based guessers (i.e., JTR-markov and OMEN) struggle, though OMEN outperforms JtR-Markov by 5%. As both guessers perform a more exhaustive search of the password space than other models, they are less suited to this task.

For three-hundred million guesses, PCFGv4 performs the best, with a 6% lead over the second best guesser Sem. The Identity guesser performs surprisingly well, with an average of 30.5% (but a high standard deviation of 14.079%) with at most 21,653,268 guesses compared to 300 million guesses for other guessers.[2] In its best

[2]The upperbound for number of guesses in Identity guesser is derived from maximum number of unique passwords in our datasets.

case, the Identity guesser trained on Twitter guesses 56.7% of RockYou, only 10.14% lower than the best guesser PCFGv4 on that same pair. OMEN under-performs JtR-Markov, performing worst overall at this cutoff.

5.2.2 Impact of Training Data Choice

We investigate how guessing success rates are impacted by different aspects of training datasets such as training dataset size, and the similarity between training and testing datasets. We also investigate how similarities between password datasets are preserved in their corresponding password guess lists. We train all six password guessers on each of the six individual plaintext datasets and test them against every other plaintext dataset, yielding 180 password cracking scenarios. For all guessers, we set the cutoff to 300 million guesses.

Figure 5.1 captures the average success rates for various pairs of training and testing datasets. One can make two important observations: (i) while some datasets perform well as training data (e.g., Twitter, Mate1), others are much less effective (e.g., Webhost); (ii) some pairs of datasets are effective for training and testing against each other, i.e., when one dataset can train guessers well against another dataset (e.g., RockYou-Mate1 and Mate1-RockYou, ClixSense-Mate1 and Mate1-ClixSense, etc.).

Although some dataset pairs (e.g., Twitter vs. ClixSense) have similar guessing success in each direction, most other pairs (e.g., Fling vs. RockYou) have asymmetry in their guessing ability. This seemingly counter-intuitive observation might be explained by how differently guessers prioritize their generated passwords based on their training dataset and its features. These observations motivate us towards a deeper analysis of the characteristics of effective training datasets.

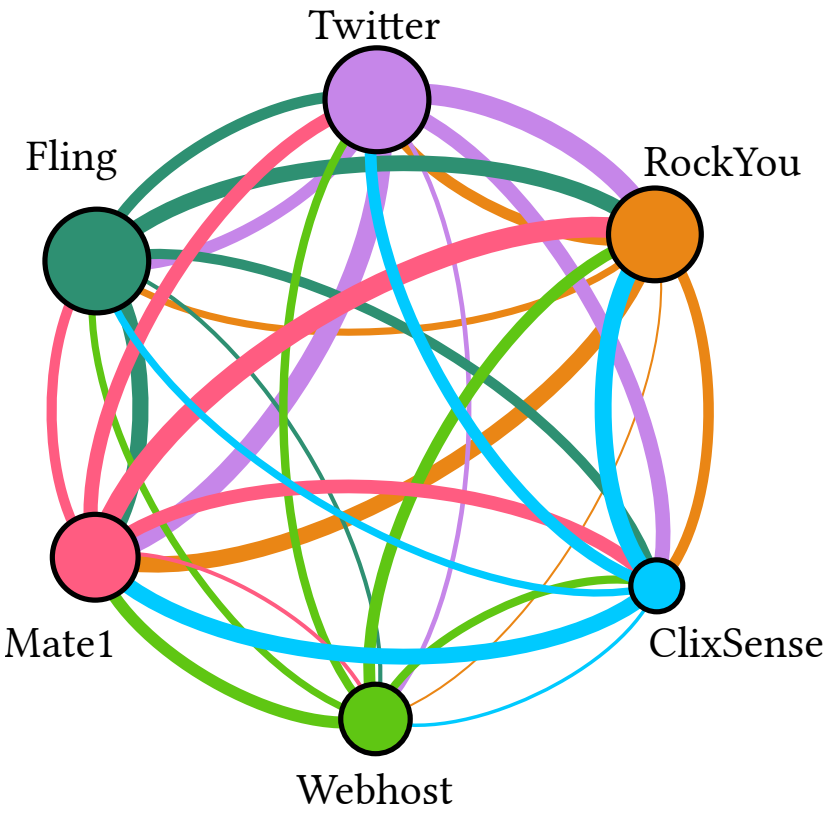


Figure 5.1: The average success rates for various pairs of training and testing datasets. Each edge width is proportional to the average success rate of all guessers for a fixed training and testing dataset. The edge colors match the training dataset color and also the outer edges are directed clockwise from training to testing dataset. The node sizes represent the size of the password datasets.

Size of training dataset

We ask whether the success rate of a guesser, on average, increases with the size of training dataset. Table 5.3 shows the average success rates of each training dataset over all guessers and target datasets (computed by Eq. 3.8), with datasets ordered from smallest to largest size. While our largest dataset performs the best, our smallest dataset ClixSense outperforms both Webhost and RockYou, which are over six and fifteen times larger than it respectively. We also note that Fling's success rate is 7.6% lower than Twitter's despite being very close in size. For a formal analysis, we

ClixSense	Webhost	Mate1	RockYou	Fling	Twitter
33.737%	29.602%	38.167%	30.264%	35.155%	42.815%

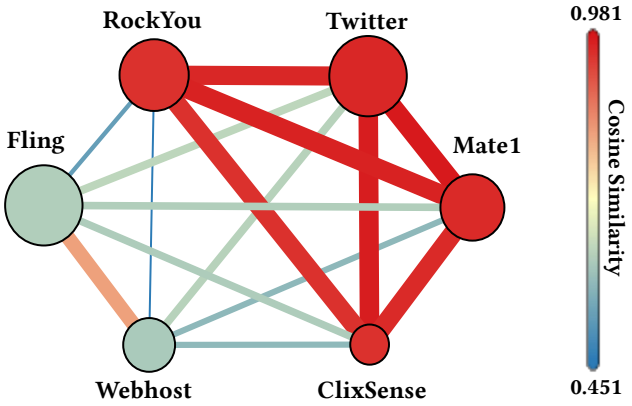
Table 5.3: Mean success rates for training password datasets. Datasets are ordered smallest to largest from left to right.

calculated the statistical correlation between the number of passwords in the training dataset and the averaged success rate. The resulting Pearson coefficient of 0.189 (p= 0.315) suggests insignificant correlation between training dataset size and success rate. This result suggests that while size of training dataset might play a role in success rate, it is not the sole influential factor.

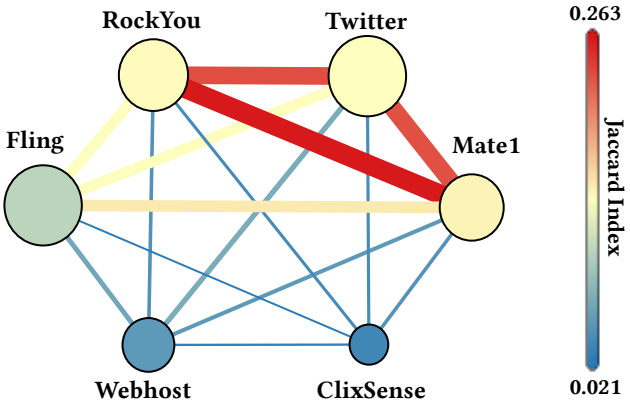
Similarity between training and target datasets

We next focus on how the similarity between training and target datasets impacts the success rate of guessers. We first compute the cosine similarity and generalized Jaccard index (see Eq. 3.2 and Eq. 3.4) between password datasets, and then explore the relationship of these similarities with success rates. This is our only experiment that can be compared with Ji et al.'s [50], in the sense their work computes similarity using cosine similarity and Jaccard index. Interestingly, although our features, datasets, and guessers differ, our results confirm their finding that training and target data similarity has an impact on guesser success rate.

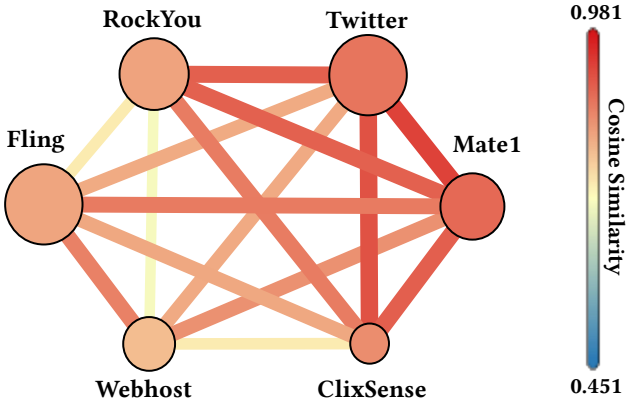
Figure 5.2a shows that that Mate1, Twitter, RockYou and ClixSense have high structural password similarity (i.e., cosine similarity). Fling and Webhost are dissimilar to other datasets, but similar to each other. These similarities between these datasets is likely why Identity performs so well in Section 5.2.1. Figure 5.2b suggests that the exact overlap between datasets (i.e., generalized Jaccard similarity) is often low with exceptions for larger datasets (i.e., Fling, Twitter, RockYou and Mate1), likely due to their size. One can draw interesting patterns by cross examining Figures



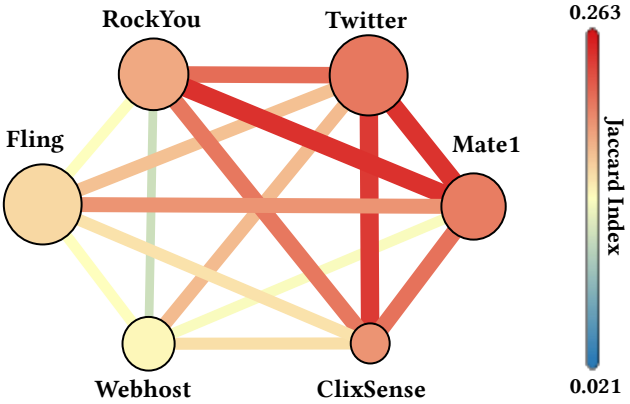
(a) The cosine similarity.



(b) The generalized Jaccard index.



(c) The cosine training similarity.



(d) The Jaccard training similarity.

Figure 5.2: Plaintext datasets with their pairwise (a) cosine similarity, (b) generalized Jaccard similarity, (c) cosine training similarity, and (d) Jaccard training similarity. The training similarity between datasets is computed by Eq. 3.12. The edge weights and colors are based on the corresponding metric value between two datasets. The node color captures the metric average for the corresponding dataset. The node size is proportional to the dataset size.

5.1, 5.2a, and 5.2b. In general, the datasets with higher similarity tend to have mutually higher success rates. For example, Mate1 and RockYou share high similarity and high mutual success rates, but Mate1 is more effective at guessing RockYou than RockYou is at guessing Mate1 despite Mate1 being smaller than RockYou.

The cross-examination of Figures 5.1, 5.2a, and 5.2b suggest the datasets with

higher similarity tend to have mutually higher success rates (e.g., Mate1 and RockYou share high similarity and mutual success rates). Thus, we hypothesize that the similarity between training and testing datasets has a positive effect on success rate. To test this hypothesis, we ran Pearson statistical tests between the similarity metric of any pair of datasets and the success rates when either dataset is used as training data to attack the other. Our cosine similarity and Jaccard metric have correlation coefficients of 0.597 ($p = 0.00049$) and 0.596 ($p = 0.00049$) respectively. Both are significant and large by Cohen's convention. This further confirms that dataset similarity, structural (cosine) or overlap (Jaccard), is a key factor in success rate. These results compliment previous findings [50] on the relationship between the similarity of training and testing datasets and guesser success rates.[3] However, it should be noted that when datasets are similar, the larger datasets usually tend to outperform smaller datasets on average (e.g., Twitter and Mate1 are highly similar, but Twitter still noticeably outperforms Mate1).

Training similarity between datasets

The surprising performance of ClixSense in Table 5.3, despite its small size, raises the question of how similarly ClixSense and a bigger dataset can train a guesser, as smaller training datasets are desirable to reduce training time. So, we next explore how similarly two datasets can train a guesser using our notion of training similarity (see Eq. 3.12). We exclude the Identity guesser due to its simplicity in learning; also, its results mirror dataset similarity (see 5.2.2) as its guesses are a type of summary of the training data.

Figures 5.2c and 5.2d demonstrate the cosine and Jaccard training similarity be-

[3]This is our only experiment with partial overlap with other work [50] by computing cosine similarity and Jaccard index between datasets; however, we not only use a different set of datasets and guessers, but also our features for cosine similarity differ, and we use Jaccard index between datasets rather than between their features.

tween our datasets. The cosine training similarity is relatively high between most pairs of datasets ranging from 0.71 to 0.93 (see Figure 5.2c). A few exceptions with moderate training similarity are Fling-RockYou, RockYou-Webhost, and Webhost-ClixSense. The Jaccard training similarity has a minimum of 0.11 for RockYou-Webhost and a maximum of 0.25 for RockYou-Mate1 (see Figure 5.2d). The cluster of RockYou, Twitter, Mate1, and ClixSense share relatively high overlap of generated passwords (see their pairwise Jaccard training similarity). This means passwords generated from training with ClixSense, despite its small size, have high overlap with passwords generated from training with other datasets. In contrast, the Jaccard training similarity between Fling or Webhost and any other datasets is relatively low, indicating that they may be better to combine with other datasets for training purposes.

Cross-examining Figures 5.2a–d, illustrates the relationship between dataset similarity and training similarity. When two datasets have low structural (cosine) or overlap (Jaccard) similarity, their training similarity is notably higher (e.g., Mate1-Webhost, ClixSense-Fling, Twitter-Webhost, etc.). However, a pair of datasets with high cosine similarity usually exhibits lower cosine training similarity (e.g., RockYou-Twitter, Twitter-ClixSense, etc.). The order of similarity usually appears to be preserved between datasets: when a pair of datasets (e.g., Fling-RockYou) have lower similarity than another pair (e.g, RockYou-Twitter), their training similarity follows the same ordering. These observations could be partly explained by guessers converging towards generating passwords with common patterns as they generalize from the training data.

5.2.3 Summary

Our analysis demonstrates that the similarity between training and testing datasets, and to a lesser extent the size of training dataset, affect the success rate of a guessing attack. These results compliment the work of Ji *et al.* [50], as we extract different features to compute structural similarity and apply Jaccard similarity directly to lists of passwords instead of their features. Our work also applies these comparisons to the guess lists generated by guessers, in addition to the datasets, and utilizes a larger set of real-world datasets. Applying Jaccard directly to the password lists grants one the benefit of seeing exact password overlap, providing context for structural similarity (e.g., two guessers might produce guesses that are similar in structure but with little exact password overlap). We demonstrate that while larger datasets result in higher success rates on average, the similarity of training and testing datasets has a larger impact on success than dataset size.

5.3 Password Guessing Behaviour

We investigate the behaviour of each guesser (i.e., their generated guess lists) under various training and target datasets. We also explore how each guesser compliments and substitutes others.

5.3.1 Training Independence

We use our training independence metric (Eq. 3.13) to study guessers' sensitivity to changes in the training dataset. Guessers with high training independence will produce similar guesses when trained on different datasets. The two extremes of training independence are a brute-force guesser with the maximum value of 1 and the identity guesser with the minimum value (not quite 0 due to overlap between training

Identity	NN	OMEN	PCFGv4	Sem	JtR-Markov
0.03783	0.11296	0.12635	0.19808	0.25595	0.38548

Table 5.4: Guessers' Jaccard training independence.

datasets). Table 5.4 reports the guessers' training independence. JTR-Markov and Identity have the most and least training independence respectively. PCFGv4, with the highest success rate, has moderate training independence. Interestingly, while OMEN and NN have similar training independence, their success rates greatly differ for both the 1 million and 300 million cutoffs. These results suggest that training independence is not a predictor of success rate, but an effective guesser (e.g., PCFGv4, NN, and Sem) has training independence in the range of 0.1 to 0.25, implying that 10%–25% of its generated passwords overlap when its training dataset is changed.

5.3.2 Generalizability

One important characteristic of guessers is how well they can generalize, i.e., predict and generate previously unseen passwords. To measure this, we train each guesser on the Webhost dataset as it is the least similar to the other datasets, both in terms of structure (see Figure 5.2a) and actual password overlap (see Figure 5.2b). We then test the Webhost trained guessers against every other dataset and calculate each guesser's mean success rate. Table 5.5 shows the mean success rate of each guesser: PCFGv4 and NN outperform others, demonstrating a relatively high degree of generalizability compared to others. The Identity guesser and OMEN perform notably worse. This is expected for the Identity guesser with its inability to generalize, but surprising for OMEN. There is a notable amount of variance in the success rates of guessers with similar approaches: 15% difference between Markov models JtR and OMEN, and 10% difference between PCFG-based guessers PCFGv4 and Sem. This highlights how even guessers with similar underlying approaches can display differing

Identity	OMEN	JtR-Markov	Sem	NN	PCFGv4
15.378%	15.664%	30.265%	33.099%	39.585%	43.618%

Table 5.5: Guessers' generalizability, with 300M guess cutoff. A higher success rate indicates a better ability to generalize.

	Training Dataset Size		
Guessers	**1 Million**	**10 Million**	**30 Million**
Identity	21.194%	33.441%	39.853%
JtR	27.570%	27.541%	27.527%
OMEN	29.077%	29.216%	29.461%
Sem	41.493%	46.910%	48.021%
PCFGv4	41.517%	48.719%	51.178%
NN	43.688%	56.500%	58.259%

Table 5.6: The mean percentage of passwords guessed by each guesser when trained on different-sized subset of Twitter with a cutoff of 300M.

generalization behaviour. When compared to their average performance at the 300M cutoff (cross-reference Table 5.2), NN experiences the lowest drop in success rate with dissimilar data (-1.2% difference), and JtR actually improves during this test (+2.7%). The guesser that performs best with dissimilar data is therefore not necessarily the one where dissimilar data causes the least impact.

5.3.3 Sensitivity to Training Dataset Size

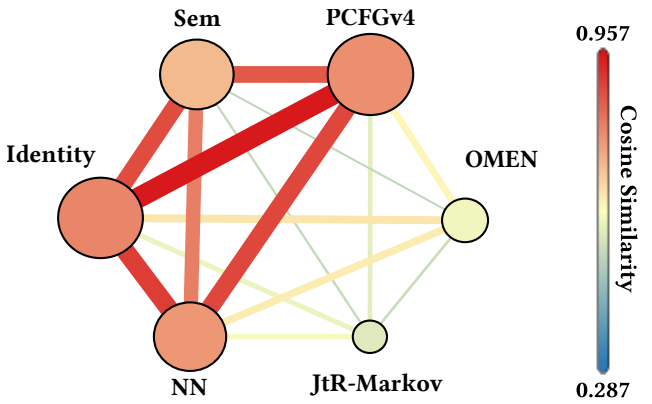
We intend to learn how each guesser's success rate is impacted by the size of training data, drawn from the same distribution. Sampling from the Twitter dataset, we create three different datasets of sizes 1 million, 10 million and 30 million. After training each guesser on each dataset, we generate guess lists at a cutoff of 300M and test them against all other datasets. Table 5.6 reports the mean success rates by Eq. 3.9. All guessers (except JtR-Markov) improve when trained on the larger dataset, but to various extents. The Identity guesser has the most drastic improvement with training size growth, from 21.2% to 39.853%. OMEN and JtR-Markov show the least improve-

ment. OMEN only improves by 0.3 percent while JtR-Markov surprisingly decreases in efficacy. Sem, PCFGv4, and NN have more modest, but notable improvements, increasing their success rates by 6.5%, 9.7%, and 14.6%, respectively. With the exclusion of OMEN, these results follow those of training independence (see Table 5.4), demonstrating that less independent guessers are typically more heavily impacted by a dearth of training data.

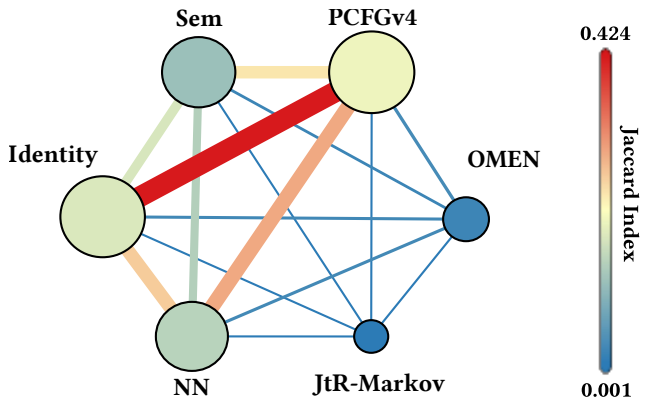
5.3.4 Guessing Similarity

Using our notion of guessing similarity (see Eq. 3.10), we analyze how similar the guess lists of two guessers are when they are trained on the same training data. Figure 5.3 shows the cosine and Jaccard guessing similarity between guessers at cutoffs of 1 million and 300 million guesses. For both cutoffs, PCFGv4, Sem, ID and NN share high structural (cosine) similarity when compared to OMEN and JtR (see Figure 5.3a and Figure 5.3c). Interestingly, despite both deploying a Markov approach, JtR and OMEN are dissimilar. As the cutoff increases (compare Figure 5.3a and Figure 5.3c), we observe that the average structural (cosine) similarity of some guessers slightly increases (e.g., OMEN, JtR-Markov) while some others slightly decrease (e.g., Identity and PCFGv4). This suggests that the relative structural similarity of passwords produced by any pair of guessers remains relatively consistent even as more guesses are produced.

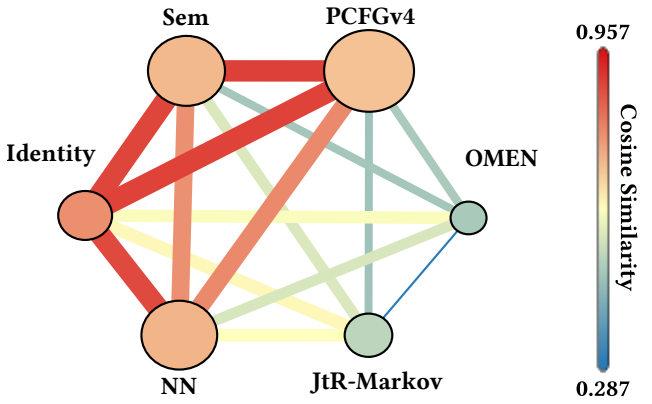
As the cutoff increases, we observe a large decrease in Jaccard guessing similarity, which captures the overlap of guessers' guesses (compare Figure 5.3b and Figure 5.3d). At a cutoff of 1 million, PCFGv4 and ID have the highest Jaccard guessing simliarity of 0.424, while at 300 million PCFGv4 and NN are the most similar pair with an index of 0.148. One can also readily observe that the Jaccard guessing similarities decrease as the cutoff increases. This change suggests that by generating more passwords,



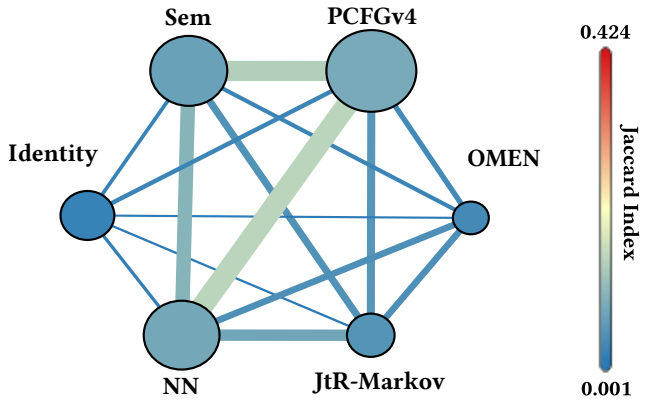
(a) Cosine guessing similarity with 1 million guesses.



(b) Jaccard guessing similarity with 1 million guesses.



(c) Cosine guessing similarity with 300 million guesses.



(d) Jaccard guessing similarity with 300 million guesses.

Figure 5.3: The cosine and Jaccard guessing similarity (see Eq. 3.10) between guessers at the cutoffs of 1 million or 300 million guesses. The edge colors represent the similarity value between two guessers. The edge width further highlights the relative similarities within a figure (thicker means more similar). The node size represents the guesser's average success rate. The node colors represent their average similarity.

each guesser converges to their own unique guessing behaviour (i.e., the percentage overlap between guessers' guess lists decreases).

We also find that guessers with higher success rate (see Table 5.2) at either cutoff tend to have higher overlap (Jaccard similarity). At 1 million, PCFGv4 and ID, the two best guessers at this cutoff, share the highest overlap. At 300 million, PCFGv4, Sem and NN achieve the highest success rates and the three highest overlaps.

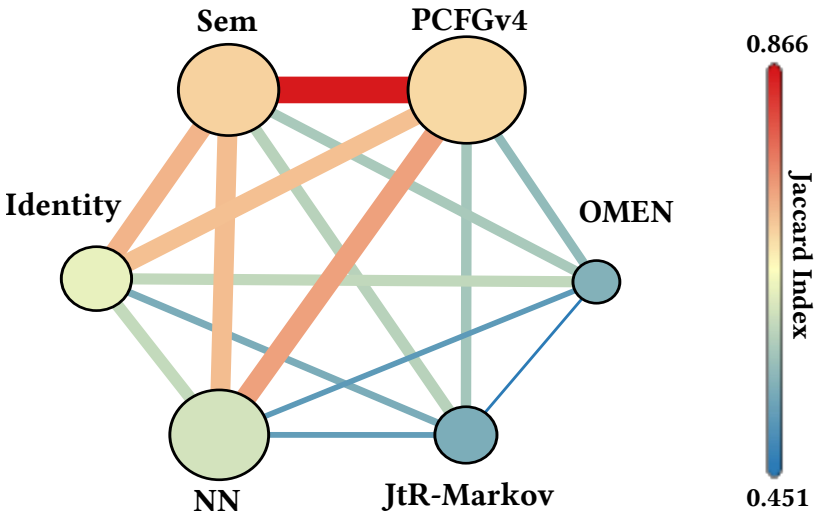


Figure 5.4: The generalized Jaccard guessing similarity between guessers for successful guesses. The edge weights and colors represent the similarity metric between two guessers. The size of the nodes represent the average success rate of the guesser. The node colors represent the average similarity of guessers with all other guessers.

At the one million guess cutoff we can observe that higher similarity to the Identity guesser is an indicator of success. This again suggests the value of training data based attacks, especially when guesses are restricted. In general, the four top guessers (i.e., PCFG, Identity, Sem and the Neural Network) share higher similarity. At three-hundred million guesses we observe that our best three guessers (i.e., PCFG, Sem and the Neural Network) share very high relative rates of structural and exact overlap similarity. Again structural similarity is high with the Identity guesser (the lower exact similarity is likely a product of the Identity guesser's low guess count). This shows that despite structure remaining the same as the cutoff increases, the actual passwords generated largely diverge.

5.3.5 Successful Guessing Similarity

Our guessing similarity analyses showed that guessers trained on the same data, generate mostly unique guesses (see Figures 5.3b and 5.3d). However, it is possible

that many of these unique guesses are unsuccessful. In this light, one might be interested in measuring the uniqueness of *successful* guesses between guessers. To achieve this, we use our successful guessing similarity (see Eq. 3.11) with generalized Jaccard index.[4]

As shown in Figure 5.4, there is still a considerable degree of uniqueness in successful guesses. Even, Sem and PCFGv4 with the highest similarity have a generalized Jaccard index of 0.86, implying that 14% of their successful guesses are unique to each other. Similarly, NN and Sem, by sharing 72% of their success guesses, owe 28% their success to their unique passwords. Interesting, the Identity guesser seems to have moderate Jaccard similarity with any other guesser (i.e., its similarity values range from 0.529 to 0.725) despite its smaller guess lists sizes (i.e., ranging from 2.2 million to 40 million compared to 300 million for all other guessers). These findings offer two important recommendations: (i) the use of one guesser does not make another guesser entirely redundant, even when the underlying approach or achieved success rates are similar; (ii) The cost-effective Identity guesser can complement any other guessers as it has a relatively high number of unique successful guesses despite it's incredibly low number of guesses comparatively. Thus, a system administrator might benefit from applying two or more different guessers for improved password checking. We explore this possibility in our combination attack discussed below in Section 5.4.

5.4 Combining Guessers

We evaluate the ability of password guessers to compliment one another on a previously unseen dataset (i.e., LinkedIn) in an offline attack scenario. We begin in Section

[4]The generalized Jaccard allows us to weight the successful guesses of each guesser based on their frequencies in the target dataset.

OMEN	JtR-Markov	Identity	Sem	PCFGv4	NN
35.641%	37.028%	47.561%	55.159%	58.798%	63.145%

Table 5.7: Percentage of LinkedIn passwords successfully guessed. Guessers are trained on the Merged dataset and cutoff at 2 billion guesses.

5.4.1 by evaluating each individual guesser against the LinkedIn dataset. Next we analyze different combinations of guessers in Section 5.4.2.

5.4.1 Individual Guessers

To compare guessers' performance, we train each guesser on the Merged dataset, and allow them to each make 2 billion guesses against the LinkedIn dataset. This experiment is most similar to the scenario when system administrators perform reactive checking. As the target dataset is unseen at the time of attack, we cannot easily tune our training data to suit it. As reported in Table 5.7, the neural network outperforms all others, with a 4.3% improvement over PCFGv4, the next best guesser. PCFG-based (PCFGv4 and Sem) and Identity guessers outperform Markov-based guessers (OMEN and JTR-Markov). Figure 5.5 depicts the percentage of guessed passwords over the number of guesses. JtR-Markov surpasses OMEN close to the end of the attack. Notably, PCFGv4, Identity, and NN traded places for the best guesser before Identity ran out of guesses. We next apply our findings from our successful guess similarity experiments to further improve the results using combination attacks.

5.4.2 Combination Attacks

Our analyses shed light on how guessers compliment others by generating unique successful guesses. In particular, we learn that Identity guesser not only complements every other guesser, but also often outperforms some advanced guessers. By combining these findings with the knowledge of the incredible cost effectiveness (both in

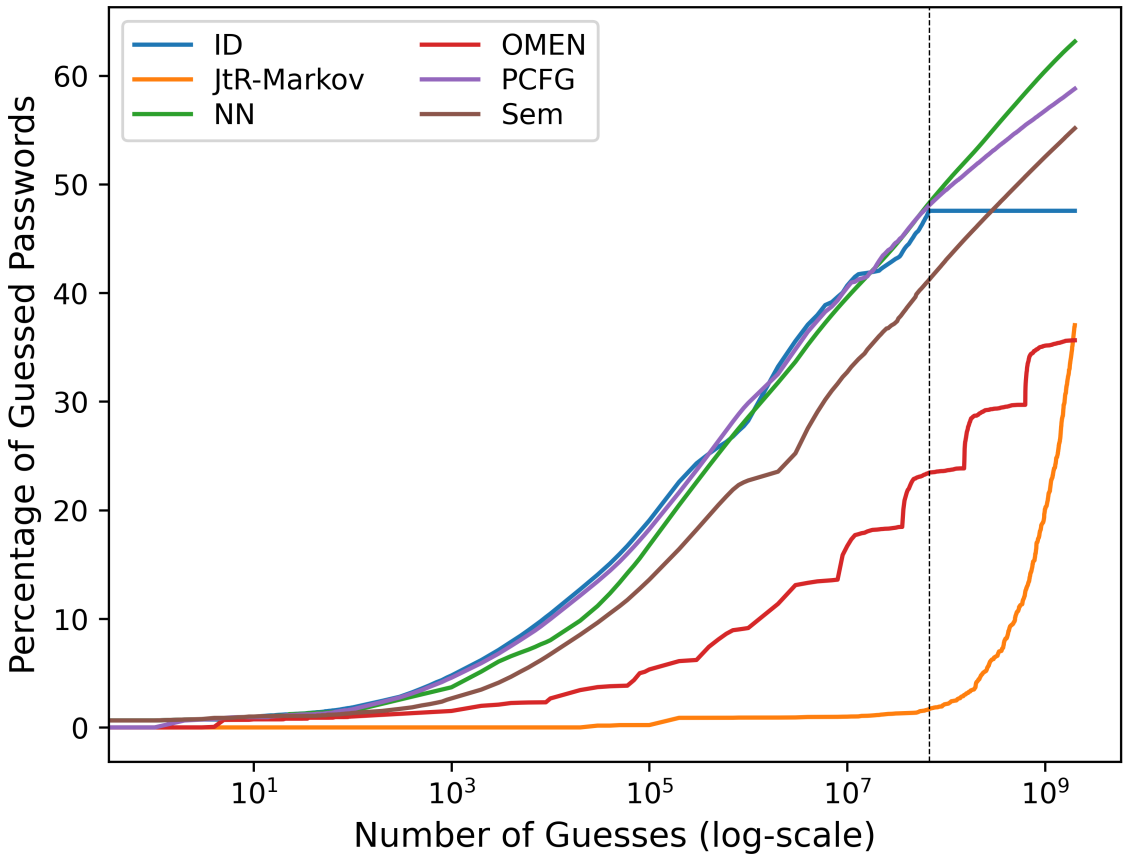


Figure 5.5: Performance of guessers trained on the Merged dataset and tested against LinkedIn. The dotted line marks the Identity guesser's last guess at 67 million guesses, each other guesser made 2 billion guesses.

terms of speed and guess numbers) we are motivated to design a *combination attack* with the Identity guesser. In this combination attack, the Identity guesser is used to attack a password dataset prior to the application of a set of other guessers. This hybrid approach exists in John the Ripper where a traditional John attack follows wordlist mode. We run many independent combination attacks on LinkedIn, where the Identity guesser precedes a subset of other guessers. Each guesser is trained on the Merged Dataset and produces two billion guesses.

The result of our combination attacks are reported in Table 5.8. When ID com-

Sole Guesser		ID + 1 Guesser		ID + 2 Guessers			
guesser	*guessed*	*guesser*	*guessed*	*guessers*	*guessed*	*guessers*	*guessed*
OMEN	35.641%	O	52.272%	O+J	57.628%	J+P	65.038%
JtR-M	37.028%	J	55.693%	O+S	61.536%	J+N	65.909%
Sem	55.159%	S	59.773%	O+P	62.907	S+P	63.866%
PCFGv4	58.798%	P	61.158%	O+N	65.241	S+N	66.411%
NN	63.145%	N	64.876%	J+S	63.255%	P+N	67.0822%

ID + 3 Guessers				ID + 4 Guessers	
guessers	*guessed*	*guessers*	*guessed*	*guessers*	*guessed*
O+J+S	63.825%	O+P+N	67.336%	O+J+S+P	66.931%
O+J+P	65.466%	J+S+P	66.614%	O+J+S+N	67.484%
O+J+N	66.199%	J+S+N	67.247%	O+J+P+N	68.060%
O+S+P	65.260%	J+P+N	67.855%	O+S+P+N	68.169%
O+S+N	66.705%	S+P+N	67.943%	J+S+P+N	68.605%

Table 5.8: The percentage of LinkedIn passwords cracked by an offline attack using the Identity guesser followed by a combination of guessers, each making two billion guesses. The names of guessers are shortened to their first letters: (P)CFG, (O)MEN, (N)N, (S)em, and (J)tR-Markov. Each combination attack is color-coded by its sequential runtime for training and guess generation of its guessers: Green is less than 8 hours (i.e., a workday), yellow is less than 16 hours, and red is over two weeks.

bined with any individual guesser (e.g., ID+O, ID+J, etc.), the combination attacks experience a notable degree of improvement compared to an individual guesser's performance (compare the columns of sole guesser vs. ID + guesser in Table 5.8). JtR-Markov experiences the largest improvement of 18.67%. Even guessers with high success rates (e.g., NN and PCFGv4) realize improvements of 1% to 4%. By dramatically increasing the success rate of weaker guessers (e.g., OMEN and JTR-Markov), this combined approach makes less resource intensive guessers more competitive.

As shown in Table 5.8, when more guessers are combined with the Identity guesser, the success rate increases, but with diminishing returns. For example, compare J to J+S (+7.562%), J+S to J+S+P (+3.359%), and J+S+P to J+S+P+N (+1.991%). There seems to be two factors in determining which additional guesser can improve an existing combination attack the most: the success rate of the candidate guesser, and

its successful guessing similarities with each of the combined guessers. A candidate guesser with higher success rate has more potential to improve the combined guesser (e.g., compare O+J to O+S). However, the candidate guesser with low successful guessing similarities can be a more effective addition. This interplay of success rate and successful guessing similarities might make a less successful guesser with lower successful guessing similarities more attractive. For example, the weaker JtR and stronger Sem have successful guessing similarities of 0.675 and 0.902 to PCFGv4. The addition of JtR to the combination attack of ID+P offers more improvement than the addition of Sem (3.88% vs. 2.71%).

Each additional guesser also incurs higher runtime and resource requirements. The attacks color-coded green in Table 5.8 could be completed within one workday (or 8 hours), whereas the yellow and red color-coded attacks must be run overnight (within 8-16 hours) and over two weeks, respectively. The neural network is the largest contributor to runtime in our combinations and also adds GPU requirements. Interestingly, unlike the sole guesser attacks, the slower combination attacks don't always outperform the faster attacks. For example, the O+J+P attack (65.466%) runs in under 8 hours while N (64.875%), O+N (65.241%), S+P (63.866%) and O+S+P (65.250%) take between 13 hours to 2 weeks. This result implies that competitive success rates can be achieved by the combination of computationally-cheap guessers with less resources. These combination attacks serve as a competitive alternative for system administrators without access to GPU resources, or with time constraints to perform reactive checking (e.g., J+P attack outperforms the neural network while running within a workday and without GPU resources).

Chapter 6

Discussion

Our work provides a number of practical recommendations (R1-R5) for system administrators auditing their password databases. While our work can be directly applied to reactive checking, it has a natural extension to proactive checking, as each of the guessers we explored can be used to generate probability scores for a given password and applied as password meters.

R1: Try publicly-available leaked passwords as a guess list. Our results show that an attacker can be relatively successful by applying Identity guesser (i.e., the training data of leaked passwords as a guess list) before considering any advanced guessers. For online attacks, the Identity guesser along with PCFGv4 outperform more advanced guessers. For offline attacks, the Identity guesser performed surprisingly well; with only 22 million guesses, on average it achieved 64% of the success rate of the top offline guesser PCFGv4 with 300 million guesses (see Table 5.2). Additionally, in our LinkedIn experiments, the Identity guesser, with 78 million guesses, had 75% the success rate of the top guesser, with 2 billion guesses (NN, see Table 5.7). These experiments strongly suggest that the Identity guesser can achieve high guessing success rates, which are comparable to the top guessers and use at least an

order of magnitude fewer guesses. Additionally, using the approx. 78 million guesses of the Merged Dataset, it was able to guess 75% the number of LinkedIn passwords as the top LinkedIn guesser in 2 billion guesses (NN, see Table 5.7). Thus, we strongly recommend that leaked password datasets should be the first priority in password checking (and ideally used as a blacklist). For this purpose, services such as *Have I Been Pwned* [45] can be useful.

R2: Apply combinations of guessers. Our results for guessing similarity show that the majority of guesses produced by each guesser are unique, even when the underlying approach or success rate is similar. Even for *successful* guesses, each tested guesser is able to crack passwords that others overlook (e.g., Identity guessed millions of LinkedIn passwords overlooked by other guessers). Our analysis indicates that no single guesser is able to completely substitute another, and when used together, which guessers compliment each other best. We also show how some combinations of guessers can have comparably high success rates with lower computing requirements. For example, in less than 8 hours and without GPU resources, Identity + PCFGv4 + JtR-Markov can achieve a success rate that compares to Identity + NN (which takes about 2 weeks and requires GPU resources). Considering both success rate and computing requirements, our results from targeting LinkedIn passwords suggest that a reasonable strategy is to apply this ordering of guessers: Identity, PCFGv4, JtR-Markov, Sem, OMEN, NN.

R3: Train with datasets similar to target. Our results show that when choosing training data, the similarity to the target data is an important factor. Thus, our metric of dataset similarity can be used to decide on the most effective training dataset. While our proposed statistics can be applied using any metric of similarity (e.g., Jaccard index), or set of features, we chose to use actual passwords and policy-

based features. As such, our results reflect similarity based on the actual passwords and policy-level characteristics (i.e., number of character classes and length), so a reasonable shorthand is to select a training dataset with a matching password policy. System administrators should therefore attempt to match their training data to the policies that exist in their own systems to maximize similarity and improve the success rate of their attacks

R4: Consider using less training data. Using more training data takes more computing resources, both in terms of longer training times and data storage. Our results indicate that training dataset size does not correlate with guessing success rates. Although when sampling from the same dataset (Twitter), we observed that data size can increase training effectiveness, the gains between 1 million and 30 million training passwords are not as large as one might expect. Therefore, if time or space constraints exist, a reasonable compromise would be to use a sample of training data from a dataset with high similarity (such as Twitter in our experiments).

R5: Use generalizable guessers if unsure about similarity. When there is considerable uncertainty of the similarity of training data, a better approach would be to use the most generalizable guessers. Our results indicate that the guessers ordered from the highest to the lowest generalizablility are: PCFGv4, NN, Sem, JtR-Markov, OMEN, and lastly Identity.

Chapter 7

Conclusions and Future Work

We provide an in-depth analysis of password guessers, revealing insights regarding their behaviours and capabilities, and when and how to use them (both alone and in combination). This information provides insight into some of the underlying factors that affect the behaviour of different password guessers. Comparing guessers against each other allowed us to understand their differences and relative similarities.

We leverage this knowledge to construct combination attacks which can strengthen weaker guessers and provide alternatives for system administrators working with limited time or resources. This work also demonstrates that combinations of computationally-cheap guessers can be comparably effective to more resource-intensive guessers. Our work allows us to provide a set of evidence-based recommendations to system administrators who use password checking tools. To assist practitioners as new guessers emerge in the future, we suggest that future research in password guessers provide runtime and resource requirements, and contrast this with the approaches to which they compare success rates.

Our proposed analytical framework (i.e., various metrics and statistics) for comparing password guessers and training datasets can be utilized or extended by practi-

tioners and researchers for future password studies. The framework is designed with metrics and statistics which could be applied to any similarity metric (i.e., they are not limited to cosine similarity or Jaccard index) or set of features. In future work, other similarity metrics and/or features can be applied for further analysis. Another interesting direction for future work is to explore how to summarize a large training dataset into a smaller dataset that trains guessers just as well. Such a smaller training dataset would decrease training time and aim to maximize success rate.

Further work may also determine how heavily impacted guessers can be made more robust to data restriction or how datasets can be tailored to perform better with highly dependent guessers. Another interesting direction would be to accurately determine a similar password dataset to use for training when various features of the target dataset are unknown. One could also look towards developing artificial intelligence algorithms to assist system administrators in finding optimal combinations of guessers with a maximum success rate under budgeted time and resource requirements.

Bibliography

[1] John the ripper community build (1.9.0-bleeding-jumbo). `https://github.com/magnumripper/JohnTheRipper`. Last Pull: July 3, 2019.

[2] The neural network password meter. `https://github.com/cupslab/neural_network_cracking`. Last Pull: May 29, 2019.

[3] Omen: Ordered markov enumerator. `https://github.com/RUB-SysSec/OMEN`. Last Pull: July 2, 2019.

[4] Pretty cool fuzzy guesser (4.0). `https://github.com/lakiw/pcfg_cracker`. Last Pull: July 15, 2019.

[5] Semantic password guesser (lite). `https://github.com/vialab/semantic-guesser/tree/lite`. Last Pull: July 16, 2019.

[6] Addas, A., Salehi-Abari, A., and Thorpe, J. Geographical security questions for fallback authentication. In *Proceedings of the 17th International Conference on Privacy, Security and Trust* (2019), pp. 1–6.

[7] Addas, A., Thorpe, J., and Salehi-Abari, A. Geographic hints for passphrase authentication. In *Proceedings of the 17th International Conference on Privacy, Security and Trust* (2019), pp. 1–9.

[8] AlSabah, M., Oligeri, G., and Riley, R. Your culture is in your password: An analysis of a demographically-diverse password dataset. *Computers & Security 77* (2018), 427–441.

[9] Baeza-Yates, R. A., and Ribeiro-Neto, B. *Modern Information Retrieval*, vol. 463. New York: ACM Press, 1999.

[10] Berkhin, P. Survey of clustering data mining techniques. In *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[11] Bhattacharyya, D., Ranjan, R., Alisherov, F., Choi, M., et al. Biometric authentication: A review. *International Journal of u-and e-Service, Science and Technology 2*, 3 (2009), 13–28.

[12] Bishop, M., and Klein, D. V. Improving system security via proactive password checking. *Computers & Security 14*, 3 (1995), 233–249.

[13] Blocki, J., Komanduri, S., Procaccia, A., and Sheffet, O. Optimizing password composition policies. In *Proceedings of the 2013 ACM Conference on Electronic Commerce* (2013), pp. 105–122.

[14] Bonneau, J. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (2012), pp. 538–552.

[15] Bonneau, J., Herley, C., van Oorschot, P. C., and Stajano, F. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (2012), pp. 553–567.

[16] Bright, P. Rsa finally comes clean: Securid is compromised. *Ars Technica* (Jun 2011). `https://arstechnica.com/information-technology/2011/06/rsa-finally-comes-clean-securid-is-compromised/`, Last Accessed: November 9, 2020.

[17] Brown, M. Unpacking the nist password requirements in 2019. *24By7Security, Inc.* (Jun 2019). `https://blog.24by7security.com/unpacking-the-nist-password-requirements-in-2019`, Last Accessed: November 9, 2020.

[18] Campbell, J., Kleeman, D., and Ma, W. Password composition policy: Does enforcement lead to better password choices? In *Proceedings of the 17th Australasian Conference on Information Systems* (2006), p. 60.

[19] Campbell, J., Kleeman, D., and Ma, W. The good and not so good of enforcing password composition rules. *Information Systems Security 16*, 1 (2007), 2–8.

[20] Campbell, J., Ma, W., and Kleeman, D. Impact of restrictive composition policy on user password choices. *Behaviour & Information Technology 30*, 3 (2011), 379–388.

[21] Castelluccia, C., Dürmuth, M., and Perito, D. Adaptive password-strength meters from markov models. In *Proceedings of the 2012 Network and Distributed System Security Symposium* (2012).

[22] Chen, B.-L., Kuo, W.-C., and Wuu, L.-C. Robust smart-card-based remote user password authentication scheme. *International Journal of Communication Systems 27*, 2 (2014), 377–389.

[23] Chien, H.-Y., Jan, J.-K., and Tseng, Y.-M. An efficient and practical solution to remote authentication: smart card. *Computers & Security 21*, 4 (2002), 372–375.

[24] Cubrilovic, N. Rockyou hack: From bad to worse. *TechCrunch* (Dec 2009). https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/, Last Accessed: November 9, 2020.

[25] Das, A., Bonneau, J., Caesar, M., Borisov, N., and Wang, X. The tangled web of password reuse. In *Proceedings of the 2014 Network and Distributed System Security Symposium* (2014), pp. 23–26.

[26] Das, S. 40 million fling.com users' passwords, sexual preferences stolen. *Hacked* (May 2016). https://hacked.com/40-million-fling-com-users-passwords-sexual-preferences-stolen/, Last Accessed: June 15, 2020.

[27] Databases Today. twitter.7z. https://web.archive.org/web/20190707133719/https://databases.today/search-nojs.php, 2019. This is an archived version of the now defunct website, close to the date the dataset was retrieved.

[28] Daugman, J. How iris recognition works. In *The essential guide to image processing*. Elsevier, 2009, pp. 715–739.

[29] de Carné de Carnavalet, X., and Mannan, M. From very weak to very strong: Analyzing password-strength meters. In *Proceedings of the 2014 Network and Distributed System Security Symposium* (2014), pp. 23–26.

[30] Dunham, M. H. *Data Mining: Introductory and Advanced Topics*. Pearson Education India, 2002.

[31] Dürmuth, M., Angelstorf, F., Castelluccia, C., Perito, D., and Chaabane, A. Omen: Faster password guessing using an ordered markov enumerator. In *Proceedings of the 2015 International Symposium on Engineering Secure Software and Systems* (2015), pp. 119–132.

[32] Dürmuth, M., Castelluccia, C., Chaabane, A., and Perito, D. When privacy meets security: Leveraging personal information for password cracking. *arXiv preprint arXiv:1304.6584* (2013).

[33] Enzoic. 3 key elements of the nist password requirements for 2020, Apr 2020. `https://www.enzoic.com/nist-password-requirements/`, Last Accessed: November 9, 2020.

[34] Florencio, D., and Herley, C. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web* (2007), pp. 657–666.

[35] Florêncio, D., and Herley, C. Where do security policies come from? In *Proceedings of the 6th Symposium on Usable Privacy and Security* (2010), pp. 1–14.

[36] Fox-Brewster, T. 13 million passwords appear to have leaked from this free web host. *Forbes Magazine* (Aug 2017). `https://www.forbes.com/sites/thomasbrewster/2015/10/28/000webhost-database-leak/`, Last Accessed: November 9, 2020.

[37] Frakes, W. B., and Baeza-Yates, R., Eds. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., 1992.

[38] Furnell, S. Assessing password guidance and enforcement on leading websites. *Computer Fraud & Security 2011*, 12 (2011), 10–18.

[39] Goodin, D. 6.6 million plaintext passwords exposed as site gets hacked to the bone. *Ars Technica* (Sep 2016). https://arstechnica.com/information-technology/2016/09/plaintext-passwords-and-wealth-of-other-data-for-6-6-million-people-go-public/, Last Accessed: November 9, 2020.

[40] Guo, Y., Zhang, Z., and Guo, Y. Optiwords: A new password policy for creating memorable and strong passwords. *Computers & Security 85* (2019), 423–435.

[41] Hackett, R. Linkedin lost 167 million account credentials in data breach. *Fortune* (May 2016). http://fortune.com/2016/05/18/linkedin-data-breach-email-password/, Last Accessed: November 9, 2020.

[42] Hanamsagar, A., Woo, S. S., Kanich, C., and Mirkovic, J. Leveraging semantic transformation to investigate password habits and their causes. In *Proceedings of the 2018 Conference on Human Factors in Computing Systems* (2018), pp. 570:1–570:12.

[43] Hitaj, B., Gasti, P., Ateniese, G., and Perez-Cruz, F. Passgan: A deep learning approach for password guessing. In *International Conference on Applied Cryptography and Network Security* (2019), pp. 217–237.

[44] Houshmand, S., Aggarwal, S., and Flood, R. Next gen pcfg password cracking. *IEEE Transactions on Information Forensics and Security 10*, 8 (2015), 1776–1791.

[45] Hunt, T. Pwned passwords. https://haveibeenpwned.com/Passwords. Last Accessed: May 21, 2020.

[46] Inglesant, P. G., and Sasse, M. A. The true cost of unusable password policies. In *Proceedings of the 2010 Conference on Human Factors in Computing Systems* (2010), pp. 383–392.

[47] Jain, A. K., and Li, S. Z. *Handbook of Face Recognition*. Springer, 2011.

[48] Jain, A. K., Ross, A., and Pankanti, S. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security 1*, 2 (2006), 125–143.

[49] Jakobsson, M., and Dhiman, M. The benefits of understanding passwords. In *Mobile Authenticatio*. Springer, 2013, pp. 5–24.

[50] Ji, S., Yang, S., Das, A., Hu, X., and Beyah, R. Password correlation: Quantification, evaluation and application. In *Proceedings of the 2017 IEEE Conference on Computer Communications* (2017), pp. 1–9.

[51] 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. *IEEE Transactions on Dependable and Secure Computing 14*, 5 (2017), 550–564.

[52] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and Julio, L. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (2012), pp. 523–537.

[53] Kiesel, J., Stein, B., and Lucks, S. A large-scale analysis of the mnemonic password advice. In *Proceedings of the 2017 Network and Distributed System Security Symposium* (2017).

[54] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[55] Komanduri, S. Modeling the adversary to evaluate password strength with limited samples (doctoral dissertation). *Carnegie Mellon University* (2016).

[56] Komanduri, S., Shay, R., Kelley, P. G., Mazurek, M. L., Bauer, L., Christin, N., Cranor, L. F., and Egelman, S. Of passwords and people: Measuring the effect of password-composition policies. In *Proceedings of the 2011 Conference on Human Factors in Computing Systems* (2011), pp. 2595–2604.

[57] Li, Z., Han, W., and Xu, W. A large-scale empirical analysis of chinese web passwords. In *Proceedings of the 23rd USENIX Security Symposium* (2014), pp. 559–574.

[58] Ma, J., Yang, W., Luo, M., and Li, N. A study of probabilistic password models. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy* (2014), pp. 689–704.

[59] Ma, L., Tan, T., Wang, Y., and Zhang, D. Personal identification based on iris texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 12 (2003), 1519–1533.

[60] MacRae, B., Salehi-Abari, A., and Thorpe, J. An exploration of geographic authentication schemes. *IEEE Transactions on Information Forensics and Security 11*, 9 (2016), 1997–2012.

[61] Malone, D., and Maher, K. Investigating the distribution of password choices. In *Proceedings of the 21st International Conference on World Wide Web* (2012), pp. 301–310.

[62] Matsumoto, T., Matsumoto, H., Yamada, K., and Hoshino, S. Impact of artificial" gummy" fingers on fingerprint systems. In *Proceedings of Optical Security and Counterfeit Deterrence Techniques IV* (2002), pp. 275–289.

[63] 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. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security* (2013), pp. 173–186.

[64] Melicher, W., Ur, B., Segreti, S. M., Komanduri, S., Bauer, L., Christin, N., and Cranor, L. F. Fast, lean, and accurate: Modeling password guessability using neural networks. In *Proceedings of the 25th USENIX Security Symposium* (2016), pp. 175–191.

[65] Mourouzis, T., Pavlou, K. E., and Kampakis, S. The evolution of user-selected passwords: A quantitative analysis of publicly available datasets. *arXiv preprint arXiv:1804.03946* (2018).

[66] Narayanan, A., and Shmatikov, V. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 2005 ACM SIGSAC Conference on Computer and Communications Security* (2005), pp. 364–372.

[67] Paivio, A., Rogers, T. B., and Smythe, P. C. Why are pictures easier to recall than words? *Psychonomic Science 11*, 4 (1968), 137–138.

[68] Pal, B., Daniel, T., Chatterjee, R., and Ristenpart, T. Beyond credential stuffing: Password similarity models using neural networks. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy* (2019), pp. 417–434.

[69] Parish, Z., Cushing, C., Aggarwal, S., Salehi-Abari, A., and Thorpe, J. Password guessers under a microscope: An in-depth analysis to inform deployments. *arXiv preprint arXiv:2008.07986* (2020).

[70] Pollack, C. Surprising password guidelines from nist you should know. *FPA, Inc.* (Oct 2018). `https://www.fpainc.com/blog/password-guidelines-from-nist`, Last Accessed: November 9, 2020.

[71] Potter, J. Npm trends: Compare npm package downloads. `https://www.npmtrends.com/zxcvbn`. Last Accessed: October 10, 2020.

[72] Ross, A., Shah, J., and Jain, A. K. From template to image: Reconstructing fingerprints from minutiae points. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 4 (2007), 544–560.

[73] Russon, M.-A. Mate1.com hack: 27 million account passwords and emails have been leaked and sold on dark web. *International Business Times UK* (Mar 2016). `https://www.ibtimes.co.uk/mate1-com-hack-27-million-account-passwords-emails-have-been-leaked-sold-dark-web-1547166`, Last Accessed: November 9, 2020.

[74] Schweitzer, D., Boleng, J., Hughes, C., and Murphy, L. Visualizing keyboard pattern passwords. *Information Visualization 10*, 2 (2011), 127–133.

[75] Shepard, R. N. Recognition memory for words, sentences, and pictures. *Journal of Verbal Learning and Verbal Behavior 6*, 1 (1967), 156–163.

[76] Singhal, A. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24*, 4 (2001), 35–43.

[77] Summers, W. C., and Bosworth, E. Password policy: The good, the bad, and the ugly. In *Proceedings of the 2004 Winter International Synposium on Information and Communication Technologies* (2004), pp. 1–6.

[78] Thomas, K., Moscicki, A., Margolis, D., Paxson, V., Bursztein, E., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., Markov, Y., Comanescu, O., and Eranti, V. Data breaches, phishing, or malware?: Understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1421–1434.

[79] Thorpe, J., MacRae, B., and Salehi-Abari, A. Usability and security evaluation of geopass: a geographic location-password scheme. In *Proceedings of the 9th symposium on usable privacy and security* (2013), pp. 1–14.

[80] Thorpe, J., and van Oorschot, P. C. Human-seeded attacks and exploiting hot-spots in graphical passwords. In *Proceedings of the 16th USENIX Security Symposium* (2007), pp. 103–118.

[81] Ur, B., Habib, H., Johnson, N., Melicher, W., Alfieri, F., Aung, M., Bauer, L., Christin, N., Colnago, J., Cranor, L. F., Dixon, H., and Emami Naeini, P. Design and evaluation of a data-driven password meter. In *Proceedings of the 2017 Conference on Human Factors in Computing Systems* (2017), pp. 3775–3786.

[82] 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. In *Proceedings of the 21st USENIX Security Symposium* (2012), pp. 65–80.

[83] Ur, B., Segreti, S. M., Bauer, L., Christin, N., Cranor, L. F., Komanduri, S., Kurilova, D., Mazurek, M. L., Melicher, W., and Shay, R. Measuring real-world accuracies and biases in modeling password guessability. In *Proceedings of the 24th USENIX Security Symposium* (2015), pp. 463–481.

[84] Van Oorschot, P. C., Salehi-Abari, A., and Thorpe, J. Purely automated attacks on passpoints-style graphical passwords. *IEEE Transactions on Information Forensics and Security 5*, 3 (2010), 393–405.

[85] Veras, R., Collins, C., and Thorpe, J. On the semantic patterns of passwords and their security impact. In *Proceedings 2014 Network and Distributed System Security Symposium* (2014), pp. 23–26.

[86] Wang, D., Zhang, Z., Wang, P., Yan, J., and Huang, X. Targeted online password guessing: An underestimated threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 1242–1254.

[87] Wei, M., and Golla, M. The password doesn't fall far: How service influences password choice. In *Proceedings of the 2018 Who Are You?! Adventures in Authentication Workshop* (2018).

[88] Weir, C. M. Using probabilistic techniques to aid in password cracking attacks (doctoral dissertation). *Florida State University* (2010).

[89] Weir, M., Aggarwal, S., Collins, M., and Stern, H. Testing metrics for password creation policies by attacking large sets of revealed passwords. In

Proceedings of the 2010 ACM SIGSAC Conference on Computer and Communications Security (2010), pp. 162–175.

[90] Weir, M., Aggarwal, S., De Medeiros, B., and Glodek, B. Password cracking using probabilistic context-free grammars. In *Proceedings of the 2009 IEEE Symposium on Security and Privacy* (2009), pp. 391–405.

[91] Weiss, K. P. Method and apparatus for positively identifying an individual, Jan. 19 1988. US Patent 4,720,860.

[92] Wheeler, D. zxcvbn: realistic password strength estimation. *Dropbox* (Apr 2012). `https://dropbox.tech/security/zxcvbn-realistic-password-strength-estimation`, Last Accessed: November 9, 2020.

[93] Wheeler, D. L. zxcvbn: Low-budget password strength estimation. In *Proceedings of the 25th USENIX Security Symposium* (2016), pp. 157–173.

[94] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. Passpoints: Design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies 63*, 1-2 (2005), 102–127.

[95] Yang, W., Li, N., Chowdhury, O., Xiong, A., and Proctor, R. W. An empirical study of mnemonic sentence-based password generation strategies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 1216–1229.

[96] Zhou, H., Liu, Q., and Zhang, F. Poster: An analysis of targeted password guessing using neural networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy* (2017).