# Evaluating Password Strength Meters and Password Composition Policies using Guessing Attacks

**David Filipe Borges Pereira**

Thesis to obtain the Master of Science Degree in

**Information Systems and Computer Engineering**

Supervisor: Prof. João Fernando Peixoto Ferreira

**Examination Committee**

**October 2020**

# Acknowledgments

My sincere thanks goes to Prof. João Ferreira, for supervising and providing me the opportunity for partnering alongside him. I am grateful for all your shared knowledge, guidance and effort put into this work without which this Master Thesis could not have been written.

In addition, a big thanks to professors Saul Johnson and Alexandra Mendes for all the efforts and time spent in the construction of this work.

Moreover, I would like to praise all the caring and encouragement that my parents gave me in order to chase my own dreams and for providing me all the means and conditions they could, enabling me to explore my academic endeavours.

For my deepest friends, I would like to thank all of you for your unconditional support and for being there whenever I needed the most.

A special thanks to my beloved girlfriend and best friend Carolina Neves, for being one of the corner-stones of my daily life and for your endless source of motivation!

Last but not least, to all my colleagues and professors who I came across to share the same class-room at Instituto Superior Técnico (IST) and for all the late night work shifts on school projects that we battled together... my sincere thanks!

Without any of you, this work would be unfinished.

# Abstract

Passwords remain the primary authentication method used in today's digital world. Despite the extensive research literature, there are still some important unresolved issues to be fixed, such as encouraging users to avoid using weak password selection behaviors combined with the reuse of credentials across different services. These behaviours make password guessing attacks a serious threat against users' accounts' integrity and their safety.

Password Strength Meters (PSMs) and Password Composition Policies (PCPs) are security mechanisms deployed with the intent of mitigating such issues, by guiding users towards better password selection and, thus, protecting them against guessing attacks. While recent studies have shown the efficacy of these mechanisms, rigorous data and new evaluation methods are needed in order to better assess their accuracy on strength estimation and overall effectiveness against this kind of attacks.

In this thesis we extend previous research literature by using well-defined evaluation methodology for studying the relationship between PSMs and PCPs with respect to their resistance against off-the-shelf guessing attacks. We show that this methodology provides a quality measure for the accuracy and effectiveness of current PSMs and PCPs under study and also highlight relevant findings about these security mechanisms. Finally, and by leveraging our experimental results on the zxcvbn meter in particular, we identify several issues regarding its internal strength estimation mechanism and propose some adjustments so as to further improve its accuracy at password strength estimation.

# Keywords

Authentication; Password Strength Meter; Password Composition Policy; Accuracy; Guess Resistance

# Resumo

A utilização de passwords continua a ser o principal método de autenticação usado hoje em dia. Apesar da extensa investigação, existem ainda vários problemas importantes por resolver, tais como evitar a combinação entre a previsibilidade na seleção de passwords por parte dos utilizadores e a reutilização de credenciais em serviços diferentes. Estes problemas permitem que ataques por adivinhação sejam uma potencial ameaça contra a integridade das contas dos utilizadores e sua segurança.

Estimadores de Segurança (PSMs) e Políticas de Composição de Passwords (PCPs) são dois dos mecanismos de segurança implementados cujo objectivo é mitigar estes problemas, guiando os utilizadores para melhores hábitos na seleção de passwords e, assim, protegê-los contra ataques por adivinhação. Enquanto que estudos recentes mostraram a eficácia destes mecanismos, dados rigorosos e novos métodos de avaliação são necessários de forma a obter avaliações mais fidedignas da sua precisão na sua estimação e eficácia geral contra este tipo de ataques.

Nesta tese estendemos trabalhos anteriores através da definição de uma metodologia de avaliação para estudar a relação entre PSMs e PCPs com a sua resistência contra ataques de adivinhação fácilmente utilizáveis. Mais ainda, mostramos que esta metodologia disponibiliza uma boa medida de precisão e eficácia para PSMs e PCPs actualmente utilizados e também realçamos observações relevantes sobre estes mecanismos de segurança. Finalmente, e tirando partido dos nossos resultados experimentais relativos ao estimador do zxcvbn, identificamos vários problemas no seu mecanismo de estimação e propomos ajustes de maneira a melhorar a sua precisão na estimação de passwords.

# Palavras Chave

Autenticação; Estimador de Passwords; Política de Composição de Passwords; Precisão; Resistência Contra Ataques por Adivinhação

# Contents

# List of Figures

# List of Tables

# 1 Introduction

## Contents

Password-based authentication is widely used in today's digital world. From the great majority of online services hosted on the Internet to offline access of personal mobile devices and local computers, passwords remain ubiquitous in current digital authentication systems [2].

Despite the universal desire to replace this form of authentication due to its security shortcomings [1], it is commonly accepted that the use of passwords in most digital services will likely remain in the foreseeable future, as they provide a viable, practical and cheap way [3] for performing user authentication.

There are many well established reasons that contribute to this pervasiveness. On one hand, service providers are able to focus their attention on the development of simple and automated ways for performing user account setup, credential storing and password revocation. On the other hand, billions of users worldwide get to experience: free-of-charge, easy-to-access and simple-to-use password-based systems. Moreover, the lack of better justifiable alternatives in terms of reasonable cost, security and usability for both users and service providers, perpetuates even further today's mainstream adoption of passwords [3].

However, passwords alone do not guarantee the absence of security related issues in digital systems. In fact, since its "birth", the use of passwords gave rise to a new myriad of problems to solve and vulnerabilities to patch. The intrinsic behavior of weak user password selection behaviors [4–6] and portfolio management [7], in addition to their poor perception and misunderstanding about the password security domain and its associated attacking threat models [1, 8], are some of the main problems regarding password security.

Furthermore, users' credentials are constantly being compromised either by security breaches in vulnerable systems [9–13] that result in passwords being stolen, leaked and then subjected to guessing attacks [14–18] or by other types of password harvesting methods, such as phishing [19] and keystroke logging [20]. In particular, given the latest developments on password guessing methods [17, 18, 21, 22] by academia, combined with the distribution of open-source guessing tools (like JohnTheRipper[1] and Hashcat[2]) and the aforementioned password data leaks (which are now publicly available), a window is left open for anyone who desires to compromise passwords' security.

In fact, the problem of maximizing resistance against password guessing attacks has been widely researched in academia for the past two decades [16, 23, 24]. A greater emphasis on studying passwords' strength, namely, on how to define and quantify it, has been extensively researched with some remarkable progress [4, 17, 25, 26].

Taking into account this nuclear problem regarding weak password selection behavior, it was only natural for new password security mechanisms to emerge. As such, password strength meters (PSMs) and password composition policies (PCPs) were designed in order to assist users during password selection and, thus, improving their password resistance against possible guessing attacks. These forms

[1]John The Ripper, https://www.openwall.com/john/
[2]Hashcat, https://hashcat.net/hashcat/

of password security mechanisms rely on the idea of proactively checking password strength through estimation [27] (according to a given metric), while enforcing certain requirements [23] and offering useful feedback to users along the way.

There have been different proposals for designing such password security mechanisms over the years and many of them have been adopted by the community [2, 24]. As expected, the majority of current online services, operating systems and even password managers have applied instances of such security mechanisms in order to improve user password selection. As an example, the zxcvbn meter[3] was developed in 2012 by Daniel Wheeler at Dropbox and it is both open-source, widely deployed across many online-based services and referenced in the literature [2, 28].

Nevertheless, building an accurate PSM is a real challenge. Depending on the metrics used to estimate passwords' strength, meters can misjudge (by inaccurately over or underestimating) the true strength of passwords, thus failing to capture its guessing resistance [2, 29]. This faulty behavior might actually misguide users into worse password selection instead of the other way around. Having this in mind, previous research focused exclusively on evaluating the impacts of current PSMs and PCPs in real-world scenarios is rather scarce [2, 24, 29], but adds crucial supporting evidence on how to develop better password security mechanisms and provide better feedback guidance towards a more secure user password selection in today's digital world. For reference, Golla and Dürmuth [2] took a step further by formulating a systematic methodology proposal for measuring and comparing the accuracy of PSMs, while providing an extensive comparison between a broad selection of meters being actively employed in a wide range of services.

Taking into account the aforementioned threats and their inflicted harm towards users' credentials being stolen, guessed and monetized [3], this thesis' motivation is focused on better understanding the relationship between guessability and current password security mechanisms being currently used.

To achieve this, we defined a precise methodology where publicly leaked password datasets are evaluated and filtered according to different PSMs and PCPs and then matched against their respective guessing resistance through the use of off-the-shelf offline-guessing attacks, in order to better assess their accuracy on strength estimation and overall effectiveness.

By following this methodology, a new collection of results was gathered thereby providing a clearer picture regarding the relationship between guessability and these password security mechanisms and extending past password research with new supporting evidence and relevant insights.

Finally, our results were also leveraged in order to suggest further improvements to the widely deployed zxcvbn meter. We identified several issues regarding its internal strength estimation mechanism and proposed some adjustments in order to improve its accuracy against guessing resistance.

[3]zxcvbn: `https://github.com/dropbox/zxcvbn`

## 1.1 Research Questions

As previously stated, the main objective of this thesis is to better understand the impacts of current password security mechanisms, namely, PSMs and PCPs, through their relationship with password guessing resistance. We have thus defined the following research questions (RQs) that we set out to answer:

**RQ1**: Does password guessing resistance to off-the-shelf attacks of similarly labelled passwords relate to their password strength estimated by PSMs?

**RQ2**: Which PCPs are the most vulnerable/resilient against off-the-shelf guessing attacks? And how are they related to PSMs?

**RQ3:** Is it possible to extract new insights from the obtained results in order to build password security mechanisms with better strength estimation and accuracy?

RQ1 is concerned with studying and understanding the accuracy of PSMs by relating how meters label passwords and how those same passwords withstand against guessing attacks. Our aim is to evaluate meters' accuracy and derive new insights from it.

RQ2 is about investigating which PCPs are the most and least reliable against off-the-shelf password guessing attacks, and thus, fit for deployment in high-security contexts. Moreover, we want to analyze how passwords filtered according to different PCPs are evaluated by PSMs in order to establish a relationship between these two password security mechanisms.

Finally, RQ3 intends to interconnect the previously gathered results with current password security mechanisms, by both analyzing and improving their strength estimation measurements towards better accuracy and protecting password-based authentication against weak password selection and possible offline guessing attacks.

## 1.2 Objectives

In order to answer these previously defined research questions, we defined a precise methodology and conducted a set of experiments regarding these password security mechanisms, followed by an extensive analysis on the results. More specifically, we took inspiration from related literature work, namely, from Golla and Dürmuth's [2] password strength meters' overview and Ur et all [17] password guessing study, and developed a different evaluation methodology in order to extend existing knowledge about the impacts of these password security mechanisms.

Starting with a selection of a wide variety of PSMs being currently employed in popular websites, a widely-studied set of PCPs [24], a collection of publicly available password leaks and several cracking tools, the methodology consisted in:

1. Producing, for each password security mechanism, a multitude of filtered subsets containing similarly labelled passwords[4], according to their output classification for password strength estimation/policy;

2. Subjecting each password against off-the-shelf offline guessing attacks that conservatively resemble an expert attacker;

3. Cross reference each filtered subset of passwords with the corresponding password guessing resistance to produce new results for evaluation, comparison and analysis.

## 1.3 Contributions

The gathered results detailed in Chapters 4, 5 and 6 provided a better understanding on the relationship between password guessing resistance and strength estimation of both PSMs and PCPs. Additionally, the resulting analysis indicates which password security mechanisms are the most accurate at estimating passwords' strength and the ones that are the most vulnerable against password guessing attacks, and thus fit for deployment.

In summary, the main contributions of this thesis are as follows:

- Provides answers to the original defined research questions, by making use of a precise evaluation methodology to assess currently used password security mechanisms and to analyse and compare their results with previous research work;

- Suggests a set improvements to be developed in the open-source zxcvbn meter derived from our results in order to improve its accuracy against guessing resistance;

- Shares a public library of different utility Python scripts[5] that were developed in the context of this thesis, that could be leveraged or extended by the password security community.

The material presented in Chapter 4 was published in RSDA 2020 [30], the 5th IEEE International Workshop on Reliability and Security Data Analysis co-located with the 31th Annual IEEE International Symposium on Software Reliability Engineering (ISSRE 2020). An extended version with the material from Chapters 5 and 6 is currently being produced and it will be submitted in November 2020.

## 1.4 Thesis Outline

The remainder of the document is structured as follows.

[4]For example, suppose that a PSM classifies passwords as either weak or strong. We produced two filtered datasets: one corresponding to the passwords classified as weak and the other to the passwords classified as strong.

[5]GitHub Repository: `https://github.com/davidfbpereira/pws_repo`

In Chapter 2 we introduce the relevant context for this thesis, by detailing the background and related work required to understand the previously defined research questions, further problems and current solutions of password-based authentication systems. We present an overview of three nuclear topics and their relationships, namely: password-based authentication and main problems, password guessing resistance and password security mechanisms.

Chapter 3 describes in more detail the evaluation methodology used in this thesis. More specifically, we specify all the steps and tools that served as basis throughout this work regarding the design of the conducted experiments.

Chapters 4, 5 and 6 showcase our experimental results regarding each research question. More specifically, we first outline the objectives and then describe the experimental designs that were conducted. Next, we present the obtained experimental results and, finally, at the end we discuss the main takeaway insights derived from these preceding results and relate them with prior literature work.

Finally, Chapter 7 summarizes the main thesis' contributions regarding the original research questions and their respective experimental results. Finally, we detail the threats to validity involved, and we present ideas for future work in the scope of improving password security mechanisms.

# 2 Background and Related Work

## Contents

Passwords have been in use since the earliest days of computing times up to now. Alone, however, they are incapable of solving security issues regarding authentication and the safety of users' credentials. In fact, since its "birth", the use of passwords gave rise to a myriad of problems to solve and vulnerabilities to patch, such as: weak user password selection and reuse, usability concerns and guessing attacks. In parallel, password-based research as been carried out and new security mechanisms have been designed and deployed into the digital realm in order to mitigate such issues and further ameliorate users' credentials security.

In this section, we showcase the necessary background in order to understand the research problems previously defined while providing an overview of password security mechanisms, including strength meters and composition policies, whose purpose is to aid users against weak password selection and to further guarantee better password guessing resistance. Furthermore, a brief summary offline password guessing attacks is also highlighted. It is important to acknowledge that this attack vector pertains a serious threat, as more effective probabilistic password guessing methods and cracking tools are being developed in combination with dedicated hardware.

## 2.1 Password-Based Authentication and Main Problems

As the name implies, password-based authentication relies on knowledge factors (some information the user knows) based on a digital input string of characters (hereafter simply mentioned as password) shared and kept secret between the user and the service provider. The use of passwords provides a way to verify users' identity and to grant him, or her, access to a secured system.

In a standard fashion, a unique identification (such as a username or email) and a secret password are selected (within a character space formed by a combination of letters, digits and symbols) by the user during account setup, bundled together and thereafter securely stored (preferably hashed and salted).

When a user wants to authenticate itself within the system, a cryptographic hash algorithm computes the hash value from the user's entry (namely, the password) and tries to match it against the hash value stored in the password database. These credentials are stored, maintained and verified by the service provider in future authentication attempts within its system.

### 2.1.1 Summary of Password-based Authentication Problems

Despite password-based authentication being a fairly understood practice by users and widely deployed in online and offline authentication systems, it still raises many concerning problems regarding users' credentials safety [3] in terms of both confidentiality and integrity.

In a more concrete way, this method gives rise to at least three points of failure susceptible to vulnerabilities and possible attack vectors: client side (in terms of password selection and usability behaviors, malware, phishing), server side (how users' credentials are stored and maintained) and network wise (for instance through man-in-the-middle attacks, sniffing). Even though these vectors are all relevant and interconnected in some sense, this thesis' scope is focused only on user password selection behaviors and guessing resistance aspects and their associated security mechanisms.

On one hand, users need to deal with password usability issues and their inflicted burden [23]. Usually this involves actions such as: selecting passwords under some given restrictions [24], future recalling, resetting (if needed) and managing a portfolio of passwords across multiple services [7]. The rising adoption of password manager applications tries to solve some of these issues by providing relief on the cognitive burden of owning, recalling and managing multiple different passwords.

Unfortunately, and despite being a safer alternative, the majority of users haven't yet transitioned to the use of password manager applications. Some obvious reasons that might explain this behavior are: not knowing or caring about the issue at all, or the associated cost of owning a subscription (even though the existence of many costless applications with fewer features in the market), or aspects related to usability/integration of services, or lack of trust due to the exposure of software vulnerabilities that can be exploited by attackers and which are later discovered [31].

Moreover, it is worth mentioning that even if people use a password manager application, they are not immune to other types of threats such as key logging [20], phishing or even guessing attacks against their stolen master password.

On the other hand, system administrators and developers of service providers need to ensure that every sensitive/private piece of data regarding their user base, such as passwords stored in their infrastructure, is well protected (in this case, preferably hashed and salted) while remaining undisclosed to the public [1]. It is of their best interest to provide a secure password-based authentication system in order to protect their users' credentials against online and offline guessing attacks (which will be detailed in the next section) and avoid any future harm to their users' data.

### 2.1.2 Main Problems: Weak Password Selection and Re-Usage

Lets now focus on a deeper problem regarding password-based authentication, namely: the freedom given to users when selecting their own passwords during account creation. This issue has the potential to originate weak passwords [6] that are potentially easy to guess by an attacker, which could later lead to account hijacking and to the loss of data integrity with nefarious repercussions.

Past password dataset leaks and current research literature point out that users tend to select easy-to-remember, small-sized passwords, with common/predictable structural patterns/rules (such as l33t speak or keyboard walks) [4]. Furthermore, in most cases passwords are embedded with personal semantics (such as containing meaningful dates, names of sports clubs or loved ones, etc) [24, 32].

These selection behaviors are mainly influenced by users' lack of knowledge or concern about password security-related domains [8], such as not having a realistic model on how attackers try to guess passwords and not knowing what makes up a secure (hard to guess) password, giving rise to a wide range of misconceptions.

Malone and Maher [5] tried to model password selection behavior distributions by studying publicly available password leaks breached from different compromised systems. In their study, and by looking at the frequency with which passwords were chosen by users, they found two interesting findings:

1. All password frequency distributions that they took into account weren't uniform at all, but rather resembled a Zipf distribution decomposed by a number of frequently used passwords followed by a long tail of uncommon password. This hints that instead of uniformly choosing from a list of non-dictionary words, the average user was more likely to choose a password which they could easily remember, leading to certain passwords being used more frequently than others and thus, more predictable and insecure;

2. Moreover, they showed that an attacker could exploit those same password distributions yielded by the skewed choice in favour of common passwords in order to reduce the computational cost of guessing passwords. By using a large set of common passwords leaks as a source of guesses, an attacker could speedup its guessing attack by using a moderate number of guesses, over simple dictionary attacks.

As an example, we sorted the RockYou password dataset leak by frequency and plotted these on a graph (Figure 2.1) against their rank in the dataset, which resembles a Zipf distribution. Moreover, Table 2.1 presents the top-10 passwords of that same dataset.

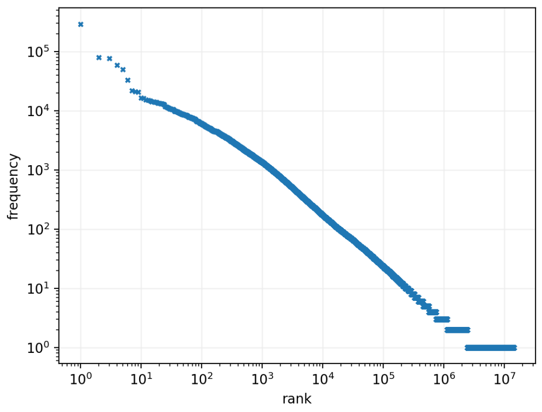


**Figure 2.1:** RockYou Dataset Rank Frequency

| Rank | RockYou | Frequency |
|---|---|---|
| 1 | 123456 | 290729 |
| 2 | 12345 | 79076 |
| 3 | 123456789 | 76789 |
| 4 | password | 59462 |
| 5 | iloveyou | 49952 |
| 6 | princess | 33291 |
| 7 | 1234567 | 21725 |
| 8 | rockyou | 20901 |
| 9 | 12345678 | 20553 |
| 10 | abc123 | 16648 |

**Table 2.1:** Top-10 Passwords of RockYou Dataset

This issue turns out to be critical for password-based systems because, in conjunction with unwanted hashed password leaks, weak password selection actually opens up a door for guessing attacks [16].

As we will showcase in the next section, anyone with internet connection and enough computing power can leverage existing open-source cracking tools to guess passwords [14, 15].

Furthermore, this problem is exacerbated by the re-utilization of weak passwords across different services. Florencio et all demonstrated this behavior by acknowledging that: as the number of accounts per user grows, reusing the same password or making small (but predictable) modifications, instead of creating new random passwords for different accounts, becomes an optimal strategy for coping with such a portfolio increase [7].

This hints that weak password selection and re-usage are behaviors that are unlikely to change, at least, for low-value accounts [1]. Even more, this behavior might be exploited in a way that when an attacker takes advantage of a compromised low-value account, he can then escalate the inflicted harm in a domino effect by trying the same password (or close variants) in other critical services.

## 2.2 Password Guessing Resistance

Password guessing is the process of recovering passwords from stolen data that has been stored in or transmitted by an authentication system. It relies on repeatedly generating candidate passwords (hereafter simply mentioned as guesses) and checking them against the available cryptographic hashes.

With the increase of the number of services being compromised/breached, resulting in millions of account credentials being leaked and stolen each year [9–13], this type of attack pertains a serious threat in the digital space and should be studied in order to be further mitigated.

Determining how well a password can withstand guessing attacks depends heavily on the following three major factors: the underlying adversary guessing model, the very strength of passwords and the guessing techniques/tools employed by the attacker.

### 2.2.1 Adversary Guessing Models

There are two different adversary guessing models [1] when dealing with ways of attacking password-protected accounts through a guessing approach: online and offline attacks.

In an online attack, the attacker tries guessing the user's credentials on a live system by spreading out the most likely password guesses over a period of time and lets the server to do the checking. This means that the attacker is bounded by the number of tries before the service provider starts flagging and blocking any further activity. In this scenario, online attacks can be mitigated, for instance, by slowing down and rate-limiting the number of guess attempts over a period of time using lockout policies [1, 24].

In an offline attack, the attacker must first gain (undetected) access to the system (or a backup) and then retrieve the stored password file, evading all back-end defences. In this case, there are four distinct possibilities concerning the storage of password files:

1. The password file data is in clear plaintext and the attacker can directly read all the passwords, with nothing to be guessed;
2. The file is hashed (preventing its direct reading) but unsalted, making it vulnerable to pre-computed password hashes lookup (rainbow) tables;
3. The file is both hashed and salted. For each guess against a user account the attacker must compute the salted hash and compare it to the stored value (guessing attack);
4. The file has been reversibly encrypted, so that its content will only be read depending on whether if the attacker has the decryption key or not.

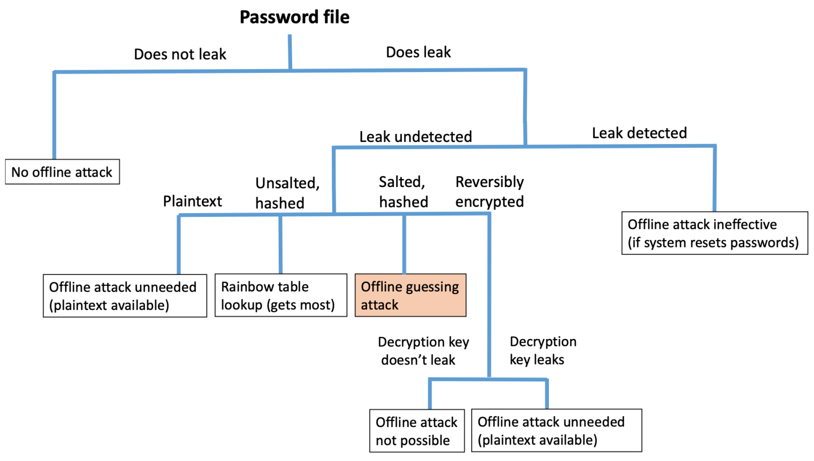

Password file
Does not leak
Does leak
No offline attack
Leak undetected
Leak detected
Offline attack ineffective (if system resets passwords)
Plaintext
Unsalted, hashed
Salted, hashed
Reversibly encrypted
Offline attack unneeded (plaintext available)
Rainbow table lookup (gets most)
Offline guessing attack
Decryption key doesn't leak
Decryption key leaks
Offline attack not possible
Offline attack unneeded (plaintext available)

**Figure 2.2:** Common Storage Practices and Offline Model Guessing Attack Path [1]

In summary, Figure 2.2 shows that offline guessing is a primary concern only when: a leak occurs, goes undetected and the passwords are suitably hashed and salted. In all other common cases, offline attack is either impossible or unneeded. If an attacker is able to do this and export the salted-hashed password file, then he is only bound by his own (offline) computational power and able to generate a number of guesses which far exceeds online attacks [1]. Moreover, password strength is now dependent on how many number of guesses each password is capable of withstanding.

Taking this into account, we focus on the offline model guessing attacks, since they pose a serious threat to passwords and are directly related to password selection behaviors. There are several off-the-shelf open-source password cracking tools specifically developed for this purpose, such as John-TheRipper[1] and Hashcat[2]. When rightly tuned, distributed (for instance with instrumented botnets or stolen cloud computing resources) and combined with powerful hardware setup (such as Graphical Pro-

[1]JtR: `https://www.openwall.com/john/`
[2]Hashcat: `https://hashcat.net/hashcat/`

cessing Units), they enable attackers to reach around 100 trillions $(10^{14})$ guesses [1, 17], roughly the same order of magnitude as the number of synapses in the human brain [33].

In addition to strengthening the security of the system itself against breaches, making passwords more computationally expensive to guess also improves the resistance against offline password guessing attacks [1, 16]. Using a computationally expensive hashing function in the authentication process to further delay each guessing attempt, such as the bcrypt[3] hashing function, is also a good security practice.

However, and as explained earlier, users tend to create predictable passwords [32] and often reuse credentials across accounts [7], making attackers' life easier even if the passwords are properly protected and stored. Therefore, the security of passwords depends directly on their relative unpredictability and strength.

### 2.2.2 Password Strength Estimation

Guessing passwords is in many ways related to password strength. In fact, trying to estimate password strength as a measure to defend against guessing attacks has a long history [2].

As recent large-scale breaches makes available significant collections of plaintext passwords and with new guessing attacks becoming more and more sophisticated nowadays, this property has also been progressively evolving throughout time.

Many metrics have been proposed and used in academia in order to measure password strength. Many of them, as it turns out, correlate poorly with guessing resistance [1, 25].

#### 2.2.2.A Older metrics

Early efforts for defining and measuring password strength were quite rudimentary and ineffective: attempting to crack hashed passwords in a given service and flagging them as weak/insecure; using certain set of heuristic rules in order to exclude weak passwords from services (marking the "birth" of password strength meters (PSMs), which will be addressed in Section 2.3); counting the number of times the same password appearing in the password database registry.

Eventually, these metrics gave way to password entropy-based metrics, which was originally defined by Shannon as the expected value (in bits) of the information contained in a string. Several entropy models for estimating passwords' strength have been used in past works [23, 32], ranging from computing the formula $L \cdot \log_2 C$, where L is the password length and C is the size of the alphabet used, to more complex predefined heuristic rules based on entropy, such as the NIST standard algorithm[4].

[3]bcrypt: https://codahale.com/how-to-safely-store-a-password/
[4]NIST: https://csrc.nist.gov/publications/detail/sp/800-63/1/archive/2011-12-12

These simple estimation techniques work well for the ideal case of random collections of characters, but are, however, completely unsuited for user-chosen passwords. As confirmed in recent password leaks, users tend to choose common words, proper nouns, predictable sequences and other structural patterns [4]. The problem with this approach is that it doesn't take into account these password selection behaviors: passwords that are far more common (thus more likely to be guessed) score much higher entropy than ones that are unique (and vice-versa). It follows that, password entropy is not a reliable/effective strength metric because it does not reflect how well passwords can withstand guessing resistance [1, 4, 16, 25].

#### 2.2.2.B Newer metrics

The failure of traditional measures to estimate password strength has led to alternatives aiming to more closely reflect how well passwords withstand guessing attacks. Thus, two main alternative guessability classes of metrics emerged:

1. Statistical metrics - models an optimal attacker with access to the actual password distribution and making guesses in likelihood order. It tries to estimate the number of guesses required for an idealized attacker to guess a fraction of a password set [4]. However, since password distributions are heavy-tailed [5], very large samples are required to determine a set's guessability accurately. Moreover, these metrics don't provide means for modeling real-world attacks;

2. Parameterized metrics - investigates guessability by focusing on the number of guesses (or the amount of computation/time needed) that a password (or set of passwords) can withstand against a properly trained and configured cracking algorithm/tool. These metrics have been used to measure password guessability in recent studies [16–18]. A set of different algorithms and widely used cracking tools are showcased in the next section.

In contrast to statistical metrics, parameterized metrics enable guessability estimation of each password individually, allowing valuable feedback during the password selection process, and in addition, to model different attacking profiles by running/simulating actual password cracking tools, rather than an idealized attacker.

However, each parameterized metric is only as good as the chosen algorithm, configuration and training data [17]. Furthermore, relying on a single out-of-the-box configuration tool underestimates a passwords guessability as it performs far more poorly when compared to a more advanced configuration.

### 2.2.3 Password Guessing Algorithms and Methods

There are many different guessing algorithms and cracking tools being developed, shared and used by attackers in order to compromise password resistance. They usually belong to one of the following approaches: heuristic or probabilistic.

The acquisition of large collection datasets of password leaks provides critical information about password selection behaviors [4, 5, 32] that can be fed into these algorithms in order to improve their efficacy in subsequent attacks.

Moreover, the emergence of hardware acceleration (with customized GPUs, FPGAs and ASICs) and distributed computation systems over the past decade also led to the increase of the efficiency and speed of current guessing attacks [14, 15] allowing billions of guesses per second.

#### 2.2.3.A Heuristic Approaches

Brute-force attacks are conceptually the simplest, but also the most inefficient guessing methods out there. Based on performing an exhaustive search of a given password keyspace, they try all possible string combinations until exhaustion. The total number of guessing attempts amounts to the alphabet size from which the characters are drawn raised to the power of the string length.

Despite being successful for targeting short passwords, as the alphabet size and string length increases the password keyspace grows exponentially, making it unfeasible for longer passwords.

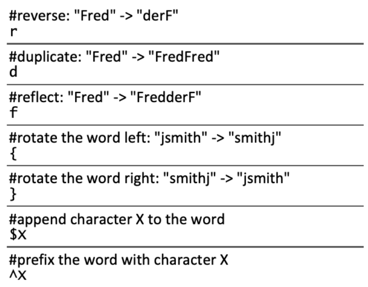
As a simple example: lets imagine we wanted to crack the password “Password1”. With an alphabet size of 62 (26 lowers + 26 uppers + 10 digits) and a password length of 9, performing a brute-force attack could take up to $62^9$ guesses in order to crack it. Imagining that we guess with a speed rate of 100 million guesses per second, it would require more than 4 years to exhaust the entire password keyspace.

Given the attackers limited time frame and hardware capacity, heuristic methods were developed in order to reduce the password keyspace to a more efficient one. Upon inspecting leaked password corpora, attackers started analyzing and mimicking common user strategies as shortcuts to generate likelier candidate guesses.

Thus, heuristic password cracking tools are most frequently used to perform what is nowadays termed as mangled-wordlist attacks. Instead of iterating over all possible string combinations, this approach applies certain transformation/mangling rules to each entry of a wordlist in order to create additional variant guesses, which are then appended back to the original wordlist.

Most wordlists contain stolen passwords from data breaches, words from dictionaries, and natural-language data, whereas rule lists specify compositions of individual transformations (including conditional logic) written in tool-specific languages. Typical examples of mangling rules are: to uppercase the first letter, append a digit to the end, replace every occurrence of the letter “a” with “@”, etc.

To guess a larger fraction of passwords more quickly, both wordlists and rule lists should be ordered in descending order of likelihood for generating successful guesses [18], since the application of each rule results in a larger wordlist and thus, in a larger number of guesses to be computed. As they gain



```
#reverse: "Fred" -> "derF"
r
#duplicate: "Fred" -> "FredFred"
d
#reflect: "Fred" -> "FredderF"
f
#rotate the word left: "jsmith" -> "smithj"
{
#rotate the word right: "smithj" -> "jsmith"
}
#append character X to the word
$X
#prefix the word with character X
^X
```

**Figure 2.3:** Basic Mangling Rules for JohnTheRipper Cracking Tool

experience, attackers augment and refine their own lists.

Golla et all presented a scientific analysis for reasoning about popular mangled-wordlist software attacks [18] by implementing a guessing-number calculator[5]. This tool efficiently determines how many guesses JohnTheRipper and Hashcat cracking tools (with a particular wordlist and rule list configuration) would issue before guessing a given password, without naively executing them.

Moreover, because word entries and rules that are included in each tool configuration have a direct impact on whether passwords are guessed at all and how quickly they are guessed, they also reasoned about optimization techniques (in terms of list ordering and completeness) for improving the efficacy and efficiency of these attacks.

#### 2.2.3.B Probabilistic Approaches

Acknowledging that not all guesses have the same probability of succeeding in a password guessing attack, academic studies focused their attention on new probabilistic guessing models where guesses are generated and assigned probabilities based on a large corpora of leaked passwords as training data.

For instance, Probabilistic Context-Free Grammar method [21] builds a model through a training set of passwords in order to form templates, that is, concatenation of sequences of letters, digits and symbols within passwords. It then calculates the probability of each password by multiplying the frequency of its templates by the frequency of each pattern in the template. This model can be used to automate the generation and assignment of probabilities to new templates and patterns, to further derive password guesses in a probabilistic order. Moreover, since this method can be trained on passwords having a particular structure, it can also be employed in cases where the target password guesses must meet special requirements and composition policies (which will be later addressed).

Recent studies of password cracking methods devised other equally effective probabilistic alternatives, such as using Markov models [22] or even neural network-based models [34].

However, while probabilistic methods perform well on a guess-by-guess basis, they impose a high computational cost for generating trillions of guesses [17]. Moreover, real-world attackers rarely use

[5]Guess-number Calculator: `https://github.com/UChicagoSUPERgroup/analytic-password-cracking`

probabilistic tools. Instead, they rely on using open-source distributed methods (like the aforementioned JohntheRipper and Hashcat cracking tools) for generating guesses.

Blase Ur et all made a significant contribution to the password security community by developing and sharing a free of charge service dubbed Password Guessability Service (PGS)[6]. It is a cloud-based service maintained by the Passwords Research Team at Carnegie Mellon University that simultaneously orchestrates several of these password guessing methods (with multiple configurations) to calculate passwords resistance.

Being more than a remote service, it also provides a way to approximately estimate passwords' strength against offline attacks resembling a guessing expert running multiple state-of-the-art guessing tools in parallel and properly configured [17].

Finally, these newly developed tools and services [17,18] can be put to use as password strength meters (discussed in the next section), by taking advantage of their password strength estimation methods and guessing resistance modeling capabilities.

## 2.3 Password Security Mechanisms

For this purpose, devising password security mechanisms to prevent users from selecting simple (really, easily guessed) passwords [5,32], while giving better feedback during the password selection process, is crucial for the security of password-based authentication.

This led to the development of password strength meters and password composition policies which will be now addressed.

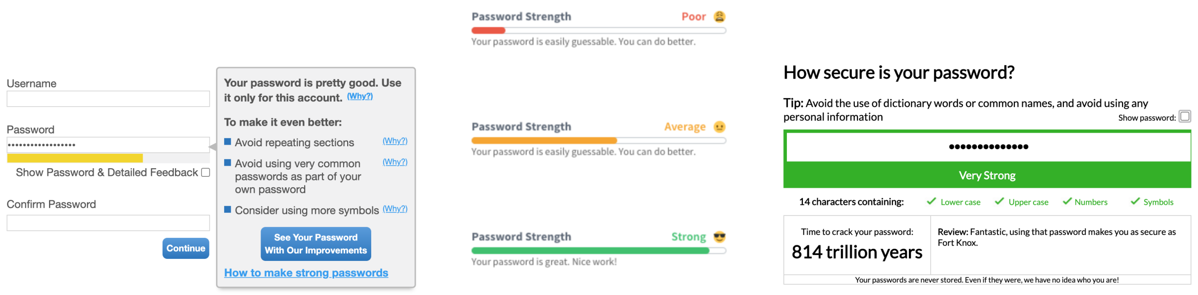
### 2.3.1 Password Strength Meters

Password checkers [27], now better known as password strength meters (PSMs) and embedded within almost all online and offline digital services [2,29], are an important tool to help users with the task of selecting safer passwords and thereby ensuring a certain level of guessing resistance beforehand.

The main function of a PSM is to provide an estimation on how strong a password selected by a user is. Put another way, we can think of it has a function that maps passwords (as input) to estimated strength score values (as output).

These estimations might be displayed by the service in countless formats and representations [2,29,35]: by using the likelihood that a password will be chosen by users, guessability or time required to guess a password, score classification based on quantization classes (strings denoting "Weak", "Medium" and "Strong", for instance) followed by a dynamic coloured strength bar (or other similar visual indicator) for a better understanding as can be seen in Figure 2.4.

[6]PGS: https://pgs.ece.cmu.edu/


Username

Password

••••••••••••••••••

Show Password & Detailed Feedback ☐

Confirm Password

Continue

Your password is pretty good. Use it only for this account. (Why?)

To make it even better:

- Avoid repeating sections (Why?)
- Avoid using very common passwords as part of your own password (Why?)
- Consider using more symbols (Why?)

See Your Password With Our Improvements

How to make strong passwords

Password Strength Poor

Your password is easily guessable. You can do better.

Password Strength Average

Your password is easily guessable. You can do better.

Password Strength Strong

Your password is great. Nice work!

How secure is your password?

Tip: Avoid the use of dictionary words or common names, and avoid using any personal information

Show password: ☐

••••••••••••••

Very Strong

14 characters containing: ✓ Lower case ✓ Upper case ✓ Numbers ✓ Symbols

Time to crack your password: 814 trillion years

Review: Fantastic, using that password makes you as secure as Fort Knox.

Your passwords are never stored. Even if they were, we have no idea who you are!

**(a)** Password Feedback (CMU) **(b)** Score Classification (Kaspersky) **(c)** Cracking Time (My1Login)

**Figure 2.4:** Various Different PSM Representations

Moreover, the strength estimation display might be coupled with valuable feedback tips or even enforce users to choose stronger passwords by filtering them with regards to exclusionary rule-sets better known as composition policies (discussed later).

There are several factors that may influence the design of an ideal PSM [29]. In order to uncover common patterns, detect weak passwords and more accurately estimate password strength, meters might employ techniques based on: heuristics [28], detection of inherent patterns and decomposition of user-chosen passwords [36], guessability of passwords [34] and other state-of-the-art probabilistic models [37].

Prior studies showed the efficacy [35, 38] of using these security mechanisms regarding passwords' strength and usability on both artificial and real-world scenarios.

More precisely, in the presence of meters users indeed changed their password selection behaviors and were motivated by either avoiding weak passwords and/or by iterating over the selection process of longer passwords while including more character classes for sensitive accounts [38]. This led to a significantly increased resistance against guessing attacks. Furthermore, it has been proven that a combination of textual and visual indicators provides a better guidance into achieving stronger passwords when compared to individual meter design decisions [35].

Finally, their impact depends heavily on the context in hands [35], meaning that for low-value accounts the meters are unlikely to influence password strength whereas for high-value accounts might have a huge impact.

### 2.3.2 Evaluating Meters' Accuracy

Given the importance of PSMs on guiding users into better password selection, one of the main properties that must be ensured is the accuracy of password strength estimation [2], that is, how well ones' password strength is correctly measured. Since most of the feedback received by users regarding password strength comes from these security mechanisms, they must strive for accuracy by giving their best estimations possible.

However, capturing the actual strength of passwords is hard in practice since the accuracy depends heavily on how password strength is defined, measured and then displayed to the user [1, 2, 25]. Failing to depict what constitutes both weak and strong passwords, might actually misguide users into worse password selection.

For instance, if easily guessed passwords are rated as being strong (thus overestimating strength), users might get a false sense of security and end up picking weak passwords, whereas, if hard to guess passwords are rated weak (thus underestimating strength), the meter might drive away users from picking strong passwords.

Furthermore, since different PSMs might use different estimation methods [2,28,34,36,37], the same password may generate disagreements/mismatches among different meters' outputs. These inconsistencies and incoherent feedback can make these mechanisms far less effective tools [29] by cultivating a false sense of security and mistrust among users [1].

It is therefore critical to measure meters' accuracy in a fair and precise way. The most common method for this is by comparing the similarity between its output classifications with an ideal baseline reference (for instance based on count frequency of a password distribution).

A recent systematic methodology proposal presented by Golla and Dürmuth's [2] tested several similarity metrics (according to the sensitivity to monotonic transformations, quantizations and noise disturbances) in order to measure the accuracy of meters. This study showed that weighted and ranked similarity metrics performed well and seemed suitable in regards to measuring PSMs' accuracy.

Moreover, besides identifying suitable similarity metrics by using their methodology proposal, they also compared different meters, despite their different estimation methods and representation displays, against simulated online and offline attacks. They showed that recent password meters developed by academia were quite good, while some password meters deployed in online services, password managers and operating systems, failed in their task [2]. This information is essential not only for selecting accurate meters but also for improving the existing ones.

Finally, building accurate PSMs is not the only challenge being solved. There are several other issues related to the design, deployment and usability aspects of meters that are out of scope from this work such as: developing reliable, compact and portable software tools and improve the feedback mechanisms displayed to users.

### 2.3.3 Password Composition Policies

As explained earlier in this chapter, the freedom that is given to users during password creation is critical in a way that it affects their security against possible guessing attacks.

There are two relevant factors that contribute to weak, and thus, predictable passwords [4, 6]:

1. Composition/structural factors, such as using: small lengths, few character classes or an uneven distribution of character classes and patterns that are easy to identify (for instance, keyboard walks, sequences, l33t speak);
2. Semantic factors, such as: dictionary words, phrases, common expressions and using personal information, (for instance, dates, names of beloved ones, favorite sports club) that are embedded within the password.

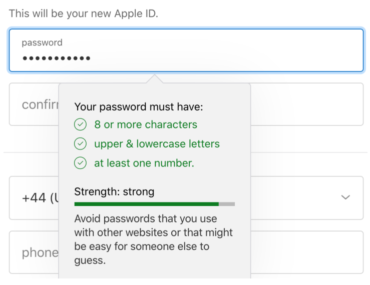
The combination of these password selection factors decreases the overall password guessing resistance that can be later exploited by the use of current guessing attacks [5, 16]. As an example: passwords composed by common strings such as "123", "password", "qwerty" are significantly more likely to be cracked under current cracking tools [24].

Having this in mind, password composition policies (PCPs) have been deployed and embedded within PSMs to restrict the space of user-created passwords. Their aim is to mitigate the aforementioned factors, by forcing users to incorporate additional complexity into passwords, while trying to maintain their overall usability in terms of password creation, recalling, user sentiment and inflicted burden [23]. In a nutshell: composition policies apply certain requirements intended to make passwords harder to guess, and hence, more secure.

There are many ways to define and create new composition policies [25]:

1. Explicit password creation - applying policies that enable users to know upfront what are the rules and requirements needed in order to create acceptable passwords;
2. Implicit password creation - applying policies that internally (within the system) reject passwords based on the guessability of passwords and other blacklisting rules;
3. External password creation - any policy that modifies a users' password after it is created so as to add more complexity to it.

Several of these previously mentioned policies can be combined, such as: requiring a minimum length, usage of different character classes (LUDS - Lower, Upper, Digit, Symbol), blacklisting common words found in a dictionary, avoiding structural patterns (such as keyboard walks or sequences), modifying or extending passwords, etc. Figure 2.5 showcases an example of applying certain composition policies in a real-world online service.


This will be your new Apple ID.

password

•••••••••••

confir

Your password must have:

✓ 8 or more characters

✓ upper & lowercase letters

✓ at least one number.

Strength: strong

Avoid passwords that you use with other websites or that might be easy for someone else to guess.

+44 (U

phone

**Figure 2.5:** Composition Policies Applied in Apple PSM During ID creation

Nevertheless, and despite depending on the security level to be achieved beforehand in a particular system, it is crucial to understand what are the impacts of selecting certain PCPs against guessing resistance and how they interact with human factors, such as usability [23].

### 2.3.4 Impacts of Policies on Password Strength and Usability

Applying PCPs without negatively influencing user behaviors is not a trivial task. Studies conducted by Komanduri et al presented a tight trade-off between the strictness of certain policies with the inflicted burden and usability of passwords resulting from such requirements [23, 24].

Under the influence of restrictive policies, users tend to have more difficulty in creating and recalling their passwords, leading to a greater struggle, frustration and self-induced coping strategies (such as reusing or using annotation methods [32]).

As an example, policies based on character class comprehensiveness alone and dictionary checks force users to strip out any kind of semantics in their passwords, thus leading to worse usability. On the other hand, relying on length requirements alone is also not a good overall choice. Despite providing better usability, they can still lead to some easily guessable passwords, as is the case with common passphrases, composition or repetition of simple dictionary words, etc [16, 24].

Even more, when fulfilling certain composition policy requirements, users tend do so in a predictable way [32], potentially making their passwords vulnerable against guessing attacks. Two examples of this behavior go as follows:

1. When enforcing uppercase letters or symbol characters in passwords, users tend to place most them at the beginning or end, respectively, thus creating patterns that can be exploitable by an aware attacker [24].

2. When dealing with more stringent policies focused on complexity, users may use simple mangling rules (such as using l33t speak or appending (easy to memorize) number sequences to the end of the password, in order to bypass password requirements, even if in a predictable way.

However, Shay et al reported that usability and password strength are not necessarily inversely correlated [24]. Their study showed that maximizing both properties at the same time while finding balance between them is achievable. As an example, deploying a combination of multiple character class and length policy requirements results in equally strong and usable passwords.

Furthermore, despite finding that blacklisting certain common strings and requiring users to evenly distribute different character classes across their password structure results in more stringent policy requirements, they also increased guessing resistance as they reject likely passwords [16] and avoid common structural patterns [24].

Nonetheless, finding the desired balance between password strength and usability is a challenge for service providers, because while stringent requirements are more suitable for high security contexts, the effects of using them incurs in usability costs for users.

Moreover, this is were PSMs come into place, since it is their job to better guide users' during password selection. Improving their feedback mechanisms [29, 35] would also allow more burdensome requirements to be better understood and accepted by users and thus facilitate the reasoning and creation of more secure passwords.

### 2.3.5 Ecosystem Validity for Collected Passwords

Finally, another relevant aspect about studying the impact of policies is the ecosystem validity and study methodology on which those collected passwords are acquired and reasoned about. In order to simulate real-world user behavior, several past studies made use of small password sets obtained through lab study samples or through crowd-sourcing marketplaces, such as the Amazon's Mechanical Turk (MTurk) [7] [16, 23, 24, 38].

Yet, in most cases, users who participated had to role-play in a fictitious environment (for instance, in creating passwords for a fake email account). Being aware that they were under study, their behaviors might have been prone to the introduction of biases with negative impact on research conclusions.

Furthermore, since users are not dealing with real accounts, those passwords have no actual value or meaning to participants [23] and can't be generalizable. In an ideal study, the collected passwords should come from a real-world scenario composed by real users and real high-value accounts in a deployed system across different services [1]. The closest we have to this ideal situation are the password sets coming from leaks of breached services.

[7]MTurk: `https://www.mturk.com/`

# 3

# Working Methodology

**Contents**

In order to investigate and evaluate the aforementioned research questions, we outlined a working methodology that was followed throughout this work. Is is important to document and describe the decision making for our choices, so as to validate our methodology and the derived insights/conclusions from it, while facilitating the reproducibility of the achieved results.

For that matter, our objective in this section is to provide both a design overview for our methodology and a more detailed description for all the steps taken that serve as the basis for our experiments.

## 3.1 Methodology Overview

Taking into account both the password-based authentication main problems and current security mechanisms involved (detailed in Chapter 2), the following already established key ideas will be taken as precursors for this work:

- Recent systematic methodology proposal introduced with the aim of measuring meters' accuracy, as well as a collection of comparison results regarding a broad selection of PSMs compiled by Golla and Dürmuth [2] and by Carné de Carnavalet and Mannan [29] works;
- Set of PCPs researched by the password community [24] with their guessability able to be tested and validated in a real-world scenario against offline guessing attacks;
- A multitude of current open-source password guessing methods and respective cracking tools supporting different configurations that mimic/approximate an experienced real-world offline attacker [17].

The main objective for this work then is to tackle the identified research questions by using a well-defined methodology evaluation, slightly different from the ones already used in past research, in order to conduct a set of experiments.

Instead of relying on password frequency distributions from publicly available leaks as the ground truth/metric for measuring passwords' strength, our plan consists in adopting a parameterized metric focused on password guessability when dealing with next to real offline attacks. More specifically, we use two different offline password guessing attack methods: heuristic and probabilistic, with properly orchestrated and configured cracking tools used and developed in the password-cracking community and in the academic literature.

The particularity for this methodology is that, in addition to measuring passwords' guessability by subjecting them against several password guessing offline attacks, each password is also evaluated and filtered according to a broad selection of strength meters and composition policies. The aim is to relate how strength meters evaluate passwords and composition policies with their respective guessing resistance.

By matching the various password strength classifications with their corresponding guessing resistance, it is possible to determine if the meter was accurately estimating their passwords strength. For example, suppose that a huge set of passwords are deemed to be strong by a given password meter. If a guessing attack can easily guess those passwords, then this suggests that the strength meter could be improved in order to be more accurate at strength estimation.

Moreover, new insights about classification mismatches and previous comparison research between PSMs and their guessing results can be taken into account for further improving these security mechanisms.

The work methodology applied for all the experiments conducted in this work can be decomposed into the following steps (also depicted in Figure 3.1):

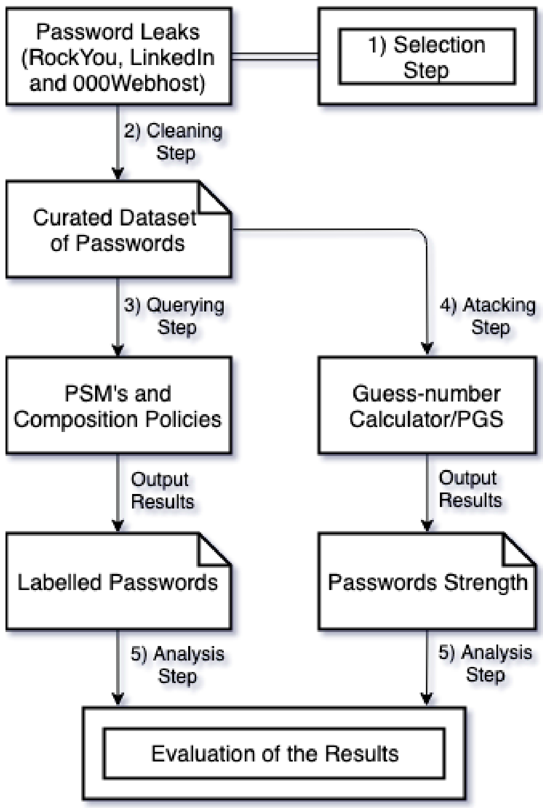


**Figure 3.1:** Work Methodology Applied in this Study

1. Selection of the PSMs, PCPs, datasets of real user chosen passwords coming from publicly available leaks and configuration of the offline guessing attack tools to be employed in the study (selection step).

2. Filtering of curated password datasets representing the real user chosen passwords to be used throughout the experiment by leveraging the password leaks previously selected (cleaning step).

3. Querying the curated datasets into smaller subsets of labelled passwords, according to the strength output classification of different strength meters and composition policies (querying step).

4. Applying offline guessing attacks against these passwords, by running open-source cracking tools which employ heuristic and probabilistic guessing methods. This will provide a conservative estimate for each password's guessing resistance when compared to a guessing expert [17] (attacking step).

5. Analysis of the results. Having each labelled password evaluated with its corresponding guessing resistance, each subset will then be compared and further assessed. As written above, the results will enable a better quantification of the guessing resistance of passwords which can then be directly compared with the strength estimation of different strength meters and composition policies, and further evaluate their accuracy and classification mismatches (analysis step).

6. Finally, based on the acquired results, leverage new ideas and insights in order to extend a currently used PSM with the objective of making it more accurate, and hence, more secure. Furthermore, and by following the methodology presented in [2], it might be also possible to evaluate and assess that meter's accuracy in relation to other meters.

In conclusion, by following this methodology we were able to gather crucial data that can be used for extending, comparing and assessing the accuracy of current PSMs. This new data also helped us determine which particular PCPs are the most vulnerable against offline guessing attacks, and thus suitable for usage in high security contexts, by knowing their overall password resistance. Furthermore, the results detailed in Chapters 4, 5 and 6 also revealed new insights on improving the accuracy of currently available PSMs (like the zxcvbn meter [28], for instance) with new ideas that stand out from the experiments and which could then be later evaluated with the methodology proposed in Golla and Dürmuth's work [2].

## 3.2 Selection Step

This section describes the selection criteria regarding the set of publicly available dataset leaks, strength meters and composition policies used in this work are described in this section as follows:

### 3.2.1 Password Dataset Leaks

Golla and Gürmuth argued that password strength in a given set of passwords is highly contextual and influenced by a wide range of factors [2]. Some examples are: the incorporation of service-related semantics during password creation (such as the name of the underlying service), the compliance of certain composition policies imposed by service providers, the meters' feedback mechanisms presented to the user during password creation and the very intrinsic weak password selection behaviors. All these factors influence passwords contained in these sets and their relative strength.

In order to minimize the influence of such factors and the introduction of biased results (coming from synthetic password-collection environments, for instance) in the experiments, three different publicly available password dataset leaks were chosen to be used in this work, namely the: RockYou, LinkedIn and 000WebHost datasets.

A brief overview of each dataset is given next and in Table 3.1:

- RockYou - compromised in plaintext from the RockYou online gaming service of the same name around the year 2009, this is the oldest and possibly the most popular password leak used in password research. It encompasses roughly 32.6 million plaintext passwords.

- LinkedIn - compromised from the professional social networking site LinkedIn around the year 2012. The true extent of this breach was unknown until 2016 when it was revealed to be much more extensive than was initially made public [39]. 172.4 million unsalted password hashes in SHA-1 format were compromised and $\approx 98\%$ of these have subsequently been cracked. These cracked passwords make up the LinkedIn dataset we use in this work.

- 000WebHost - compromised from a free web space provider for PHP and MySQL applications. The data breach became public in October 2015. The version we obtained contains approximately 15 million passwords. Based on the official statement, a hacker breached the server, by exploiting a bug in an outdated PHP version, which means that no bias was introduced.

| **Dataset** | **Year** | **Size** | **Policy**[1] |
|---|---|---|---|
| RockYou | 2009 | 32,603,048 | length $\geq 5$ |
| LinkedIn | 2012 | 172,428,238 | length $\geq 6$ |
| 000WebHost | 2015 | 15,271,208 | length $\geq 5 \wedge$ digits $\geq 1$ |

**Table 3.1:** Overview of the Password Dataset Leaks

Finally, the great majority of the passwords contained in these publicly available dataset leaks are limited to predominantly English speaking users and their password behavior preferences.

### 3.2.2 Password Strength Meters

The inclusion criteria regarding all PSMs evaluated in this work is based on two relevant factors: usability and accessibility.

[1] These were the active policies in place at the time when each breach happened as inferred by [40].

Millions (if not billions) of users interact with PSMs in their everyday life, or during their digital lifetime, as means to strengthen their password selection behaviors during account setup [2, 29, 38], while using online services. Therefore, the meter selection range included in this work should represent these commonly used online services among users.

This factor will be taken into consideration by focusing on the amount of user traffic requested in these services. Most of the meters considered in this study are from popular websites appearing in the top 100 ranking published by RankRanger[2] according to user online traffic in 2019. Each of these websites has their own version of a PSM embedded within their account creation page.

Another important factor is accessibility. We focus on meters that are easily queryable, i.e. where the setup process together with password feeding and output scraping can be automated in order to provide greater data collection efficiency in terms of both time and effort. More specifically:

- for meters with a web version interface, it should suffice the use a browser automation tool, such as the Selenium framework[3] for injecting and scraping out the necessary data;
- for meters without a web version interface, their codebase should be open-source, with proper documentation on how to setup, configure and execute the password strength meters' code while being fully-functional for our needs.

In addition, we include the HaveIBeenPwned? service[4], which is known for actively collecting database dumps with information about billions of leaked accounts and their respective passwords.

The meters that we selected for this study are shown below. Where applicable, we indicate how many bins each meter uses.

- zxcvbn (5 bins) - popular academic meter created by Daniel Wheeler [28]. We used the Python implementation[5]. It provides both the guessing number (in common logarithm with base 10) and a score estimation (within 5 bins) for a given password.
- haveibeenpwned - this web service returns the frequency of a particular passwords' hash in the available password leaked dataset collections.
- **From popular websites:**
  - 3 bins: airbnb, `airbnb.com`; bestbuy, `bestbuy.com`; thehomedepot, `homedepot.com`
  - 4 bins: dropbox, `dropbox.com`; facebook, `facebook.com`; microsoftV3, `bit.ly/39LCXT6`
  - 5 bins: cryptowallet, `blockchain.com`; reddit, `reddit.com`; slack, `slack.com`; twitter, `twitter.com`

[2]RankRanger Top 100 Websites: `https://www.rankranger.com/top-websites`
[3]Selenium Framework: `https://www.seleniumhq.org`
[4]Have I Been Pwned?: `https://haveibeenpwned.com`
[5]zxcvbn Python module: `https://github.com/dwolfhub/zxcvbn-python`

  – 7 bins: target, `target.com`;

All these online service meters represented in this selection are also part of the CCS'18 work [2], except for the cryptowallet, slack and target online meters.

### 3.2.3 Password Composition Policies

The most representative composition policies studied in current password research will also be taken into account [24]. This set is mainly composed by a mixture of length, character-class and dictionary requirements and are described as follows:

- basic8, basic12, basic16, basic20: to comply with policy basicN, passwords must have at least N characters or greater in length. No other requirements.
- 2class12, 2class16, 3class12, 3class16: to comply with policy NclassM, passwords must be M characters or greater in length and contain at least N of the four character classes (uppercase letters, lowercase letters, digits and symbols).
- 2word12, 2word16: to comply with policy 2wordN, passwords must be N characters or greater in length and consist of at least two strings of one or more letters separated by a non-letter sequence.
- comp8: password must be 8 characters or greater in length and contain uppercase letters, lowercase letters, digits and symbols. When all non-alphabetic characters are removed the resulting word cannot appear in a dictionary, ignoring case (we used the Openwall "tiny" English wordlist).
- dict8: same as comp8, but doesn't need to contain all LUDS character classes.

In addition, we also considered a couple of Linux Pluggable Authentication Modules (PAM), because they consist of software packages which enforce composition policies [41] that are used by default in widely deployed Linux systems:

- PAM_cracklib[6]: passwords must not be palindromic and be at least 9 characters long. However, passwords may be 1 character shorter if they contain at least one lowercase, uppercase, digit or symbol character. This shortening of minimum length will stack, for a minimum length of 9 - 4 = 5 for passwords containing all 4 classes.
- PAM_passwdqc[7]: passwords consisting of 1 character class only or over 40 characters long are disabled; passwords containing 2 character classes that do not meet the requirements for a passphrase, must be at least 24 characters long; passphrases, consisting of at least 3 words,

[6]pam_cracklib: `https://linux.die.net/man/8/pam_cracklib`
[7]pam_passwdqc: `https://linux.die.net/man/8/pam_passwdqc`

need to have at least 11 characters long; finally, passwords containing 3 or 4 character classes, must be at least 8 and 7 characters long, respectively. Moreover, when calculating the number of character classes, upper-case letters used as first character and digits used as last character are not counted.

## 3.3 Filtering and Sampling Steps

The filtering step can be decomposed into three sequential stages where data is first normalized, then validated and finally cleaned. The resulting dataset files contain the filtered passwords that were randomly sampled and used throughout the rest of this work. The pipeline is described as follows:

### 3.3.1 Data Normalization Stage

The first stage consists in normalizing the data contained in all original public dataset leak files that were previously gathered. The original dataset leak files' content is structured as follows:

- RockYou - text file, where each line contains the password frequency followed by the respective password (in this order).
- LinkedIn and 000WebHost - comma-separated values (csv) files, where the first column contains the password and the second column its respective frequency (in this order).

Therefore, we ended up converting the RockYou original dataset text file into a csv file by applying the same data format used in the other two original csv dataset files, thus normalizing all data and facilitating the following filtering stages.

### 3.3.2 Data Validation Stage

The next stage of the pipeline is responsible for scanning the csv files in order to remove duplicated entries contained therein. The duplicated entries are identified in Table3.2.

| Dataset | Duplicated Entries | Frequency |
|---|---|---|
| RockYou | ”johnny?”, ”????????????????”, ”????” | 1 |
| 000WebHost | ”Malas5”, ”S1*//k”, ”imranbayad7”, ”gmais01black10x0%”, ”2109”, ”M34Ghu” | 1 |

**Table 3.2:** Duplicated Entries Found In Each Dataset File

These duplicated entries were removed from the already normalized csv dataset files and their associated password entries were then updated by proportionally incrementing their respective password frequencies.

### 3.3.3 Data Cleaning Stage

The final stage of the pipeline was responsible for cleaning up passwords according to a set predefined properties (used in previous works) and originating the resulting filtered dataset files that were used and sampled throughout our work. As recommended in this type of studies [42], each dataset was first filtered according to the password composition policy it is known to have been created under [40].

Passwords containing non-ASCII characters were then removed in order to avoid encoding issues that might arise due to multi-byte characters being stored as multiple characters, artificially inflating password length. Moreover, and as already mentioned in the previous section, the great majority of the available data reflects mostly an English speaking community. To avoid processing errors during the querying step all passwords longer than 64 characters were also removed from the dataset files.

Nearly 7.5 million passwords were removed from the LinkedIn dataset file, where a small proportion (88 318 passwords) appeared to be hexadecimal data (structured as "HEX[hexadecimal number]"), while the great majority (7.4 million passwords) resembled an unfamiliar type of encoding. Moreover, passwords containing HyperText Markup Language (HTML) where also removed from the datasets.

Finally, as shown by Bonneau [4], approximating strength for unlikely passwords is error-prone. As such, each dataset was filtered once again by taking into account its own password frequency distribution, thus resulting in two separate datasets: one with relaxed conditions (without taking into account password frequency) and one with unrelaxed conditions (that only includes passwords whose frequency is at least 10).

After filtering, RockYou, LinkedIn and 000WebHost unrelaxed and relaxed datasets ended up with 43.4% and 99.7%, 36.8% and 91.3%, 12.9% and 99.8% passwords of their original dataset, respectively.

**Sampling Tools.** Finally, each password dataset file was then randomly sampled with Python's random library[8], with the Ampasamp tool[9] developed by Saul Johnson and the Python's implementation of the PAM passwdqc module[10] according to the recommendation of the Openwall Project.

## 3.4 Querying and Attacking Step

**Querying Tools.** Regarding the querying step, and given that manually acquiring all the necessary passwords' evaluations is very costly in terms of both time and effort, we decided to automate this process. Therefore, each password dataset sample file was queried using either: the Selenium framework[11]

[8]Python's random library: https://docs.python.org/3/library/random.html
[9]Ampasamp: https://github.com/sr-lab/ampasamp
[10]Python PAM passdwqc: https://alastairs-place.net/projects/pwtools/
[11]https://www.selenium.dev/

(combined with Python) or their native source-code (in the case for the zxcvbn and haveibeenpwned service meters).

More specifically, we take advantage of the Selenium framework in order to automate a headless Google Chrome browser to crawl the online meters selected for this work. The crawling is responsible for the injection of lists of passwords and scraping out their respective evaluations.

Moreover, since we were dealing with thousands of passwords in each sample, we decided to speed up this automated process by using the Python's multiprocessing library[12] in order to leverage multiple processes at once, and thus shortening the time taken to collect all the necessary data.

**Attacking Tools.** To study password guessing resistance during the attacking step we selected two different conceptual methods widely popular in the password-cracking community and in the academic literature.

We locally ran two heuristic cracking tools: JohnTheRipper (JtR, v1.8.0.9-jumbo) and Hashcat (v5.1.0). We also used three probabilistic cracking tools (Probabilistic Context-Free Grammar, Markov Model and Neural Network-based) from the Password Guessability Service (PGS) by CMU.[13]

While JtR and Hashcat cracking tools already include wordlist and rule list samples, they are far smaller than those used in typical attacks and far more ineffective [17]. Therefore we adopted an advanced configuration (rather than using the default configuration), by making use of wordlists far more extensive[14] (with near 304,000 and 1,600,000 common password entries and natural-language dictionaries). Moreover, we combined the stock (151), SpiderLabs[15] (5,146) and Megatron[16] (15,329) mangling rules for JtR and the Best64 (77), T0XIC (4,085) and Generated2 (65,117) mangling rules for Hashcat. These rule lists are often released by expert password crackers and their effectiveness has been proven in recent password research works [17, 18]. The commands used are detailed in A.1.

Finally, the PGS probabilistic cracking tools were also trained by using leaked passwords and dictionaries (as detailed in their website) in order to generate guesses. The current implementation of the Neural Network-based cracking tool only supports passwords whose length is at least 8 characters long but not greater than 31 characters. Moreover, we used their recommended configurations, as explained in [17].

**Experimental Setup.** The local experiments were performed in a MacBook Pro laptop, running macOS Catalina (version 10.15.1) with the following specs: 2,7 GHz Intel Core i5 dual-core CPU, 8 GB RAM and Intel Iris Graphics 6100 1536 MB GPU.

[12]Python's multiprocessing library: `https://docs.python.org/3/library/multiprocessing.html`
[13]PGS service: `https://pgs.ece.cmu.edu`
[14]`https://github.com/berzerk0/Probable-Wordlists`
[15]https://github.com/SpiderLabs/KoreLogic-Rules
[16]https://bit.ly/30o6Fuf

4

# PSM-Based Analysis (RQ1)

**Contents**

This chapter explores our first research question, which is on the relation between password guessing resistance and the accuracy of strength meters. Moreover, the experimental results showcased here served as the basis for a paper submitted in RSDA 2020 [30], the 5th IEEE International Workshop on Reliability and Security Data Analysis co-located with the 31th Annual IEEE International Symposium on Software Reliability Engineering (ISSRE 2020).

We will first outline the objectives regarding this research question and then describe the experimental designs that were conducted in order to answer it. Next, we present the obtained experimental results and, finally, at the end of this chapter we discuss the main takeaway insights derived from these preceding results and relate them with prior literature work.

## 4.1 Objectives and Experimental Design

The first research question that we set out to answer was:

**RQ1**: Does password guessing resistance to off-the-shelf attacks of similarly labelled passwords relate to their password strength estimated by PSMs?

This question is concerned with studying and measuring the accuracy of strength meters through the use of password guessing resistance against off-the-shelf guessing attacks.

As stated previously, PSMs are a popular password security mechanism focused on guiding users into better password selection. For that matter, they rely on different (heuristic and probabilistic) methods for evaluating/estimating passwords' strength. On top of that, widely used web services also rely on different quantization scales (in terms of both the number of bins displayed and how that scale division is performed) as means to aid users in choosing stronger passwords.

These two factors are directly linked with the accuracy of meters, as they contribute to how passwords' strength is first estimated and then displayed to users during password selection.

It is therefore important that they provide accurate depictions of what constitutes both weak and strong passwords. Meters that misjudge, by over or underestimating, the true strength of passwords might actually misguide users into worse password selection.

For instance, if easily guessed passwords are rated as being strong, users might get a false sense of security and end up picking weak passwords, whereas, if hard to guess passwords are rated weak, the meter might drive away users from picking strong passwords.

Therefore, our aim with this research question is to study the relation between meters' classifications and their respective password guessing resistance, in order to extract new insights from the obtained results and build password security mechanisms with better guessing resistance and accuracy.

To do this, we first examine how PSMs meters rate passwords while determining their password resistance against widely popular guessing attacks in the password cracking community and in the academic literature. We then check whether this relation is a reliable metric for evaluating their accuracy, while providing a comparison regarding strength estimating between a wide selection of meters.

In order to achieve this objective and, thus, answer this research question we designed the following experiment:

- First, we randomly sampled 10,000 passwords from each previously filtered RockYou, LinkedIn and 000WebHost publicly available dataset leaks (with both relaxed and unrelaxed frequency conditions[1]), having a grand total of 60,000 random passwords for this experiment;
- Second, we then evaluated those 60,000 passwords by querying the 13 meters considered in this work. Each meter will therefore have its own password classification distribution (for each dataset sample) according to its own quantization scale;
- We then set out to attack these passwords by using two different guessing methods: exploiting local heuristic attack tools (JohnTheRipper and Hashcat) and taking advantage of a remote service/infrastructure (PGS by CMU) composed of probabilistic attack tools (Markov model, PCFG and Neural Network-based methods);
- Finally, by relating password guessing resistance with their respective meters' strength classifications, it is possible to analyze each meters' password distribution in light of cracked and uncracked passwords per bin. Moreover, we are able to perform a relative comparison between the number of similar and opposite meter classifications. For example, for each cracked password in a given dataset, we can check which meters classified that same password with opposed strength labels (e.g. Weak vs Strong and vice-versa);

**Experimental Setup.** All attack tools configurations are described in Section 3.4, while the local experiments were performed in a MacBook Pro laptop, running macOS Catalina (version 10.15.1) with the following specs: 2,7 GHz Intel Core i5 dual-core CPU, 8 GB RAM and Intel Iris Graphics 6100 1536 MB GPU.

## 4.2 Experimental Results

In this section we present the results of the experiments carried out in order to analyze password strength meters' accuracy through the lens of guessing resistance.

[1]detailed in Section 3.3

### 4.2.1 Password Meter Classification Results

We start with the password classification distributions for each meter before performing the various guessing attacks. These password classification distributions were gathered after the querying step previously described in Section 3.4 in the Methodology chapter and are showcased under two different perspectives: first, according to the whole sample of 60,000 random passwords and then according to each dataset sample of 20,000 random passwords each.

These results are useful because they give us information about the estimation behaviour for each PSM without the associated password guessing resistance results (which will be introduced in the next subsections).

Figure 4.1 shows the password classification distributions for each strength meter on the whole sample of 60,000 random passwords made up of the RockYou, LinkedIn and 000Webhost datasets. Each percentage corresponds to the number of passwords under each meters' classification quantization bin.

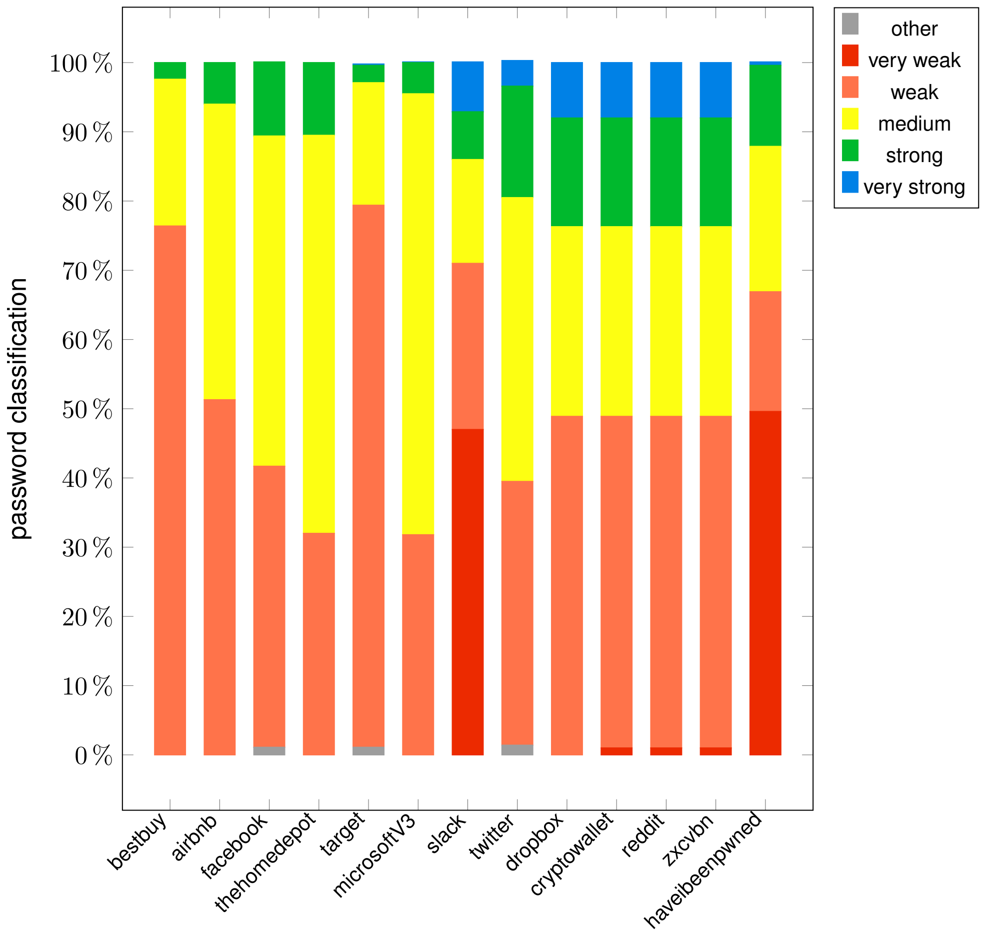


**Figure 4.1:** PSMs Classification Distributions

As anticipated, each meter has its own quantization scale composed by a small number of bins (in most cases, between 3 and 5) which are represented by different colours. The great majority of these bins are represented by a textual or numerical descriptions, where the lowest bins are commonly called “too short”, “weak” or “1 / 4”, whereas the highest bins are commonly named as “good”, “very strong” or “4 / 4”.

We clustered the categories “too short”, “too long” and “cannot contain ˜or spaces” (from the Twitter, Facebook and Target services) into one single bin dubbed “other”. We made this decision because there are few passwords with these assigned classifications and because it simplifies data analysis.

Moreover, since the HaveIBeenPwned service outputs the number of occurrences for each password, we decided to map that number to one of 5 bins based on logarithmic intervals of occurrences. In this case, a password not found on the HaveIBeenPwned database service was deemed “very strong”, 1 single occurrence was deemed “strong”, 2 to 5 occurrences deemed “Medium”, 6 to 50 occurrences deemed “weak” and over 50 deemed “very weak”. This mapping makes it easier to compare between this meter’s classification behavior and the rest.

As can be seen in Figure 4.1, Best Buy, Airbnb and The Home Depot password meter services have the fewest number of bins (composed only by “weak”, “medium” and “strong” classifications). The Facebook, MicrosoftV3 and Dropbox services have 4 bins, whereas all the others have 5 classification bins. Before we created the bin “other”, Target service was the strength meter with more bins (7).

It is also important to note that the Dropbox, Cryptowallet and Reddit services produce almost the same password classification distribution results as the zxcvbn meter (Dropbox seems to combine the two weaker bins, but maintains the remaining ones intact). This suggests that these service meters make use of the zxcvbn meter internally, thus facilitating our analysis.

Finally, we can observe four distinct meter groups, namely: conservative meters in both the lower and higher bins (Best Buy and Target services); less conservative meters in the lower bins but not the higher ones (Airbnb, Facebook, The Home Depot and MicrosoftV3 services); conservative meters in the lower bins but not the higher ones (Slack service); and less conservative meters in both lower and higher bins (Twitter, zxcvbn and its derivatives).

The quantization scale used for the HaveIBeenPwned service shows that nearly half of the randomly sampled passwords appears at least 51 times in their database, while the other half appears less that 50 times. Almost 12% were found only once and less than 0.5% were not found in their password database.

Figure 4.2 shows the password classification distribution divided into the RockYou, LinkedIn and 000WebHost dataset samples of 20,000 random passwords. In particular, it illustrates the relative classification differences between each individual datasets.

In general, the passwords from the 000WebHost dataset sample were classified as being stronger than the ones from the LinkedIn and RockYou dataset samples. The LinkedIn dataset sample also had slightly better ratings when compared to the RockYou dataset sample.

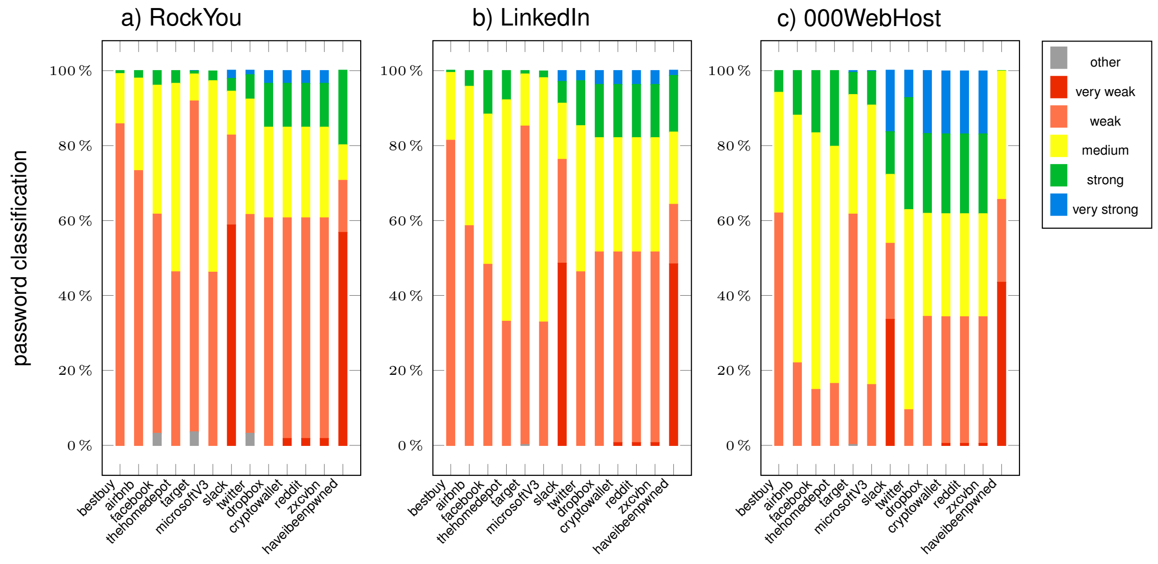


**Figure 4.2:** PSMs Classification Distributions for all Datasets

### 4.2.2 Password Guessing Attack Results

In this subsection, we determine the password guessing resistance of the 60,000 randomly sampled passwords (under each dataset) through the use of two different cracking methods: heuristic and probabilistic cracking tools.

More specifically, we properly configured the JohnTheRipper and Hashcat heuristic cracking tools (both running on local hardware) and leveraged the CMUs' PGS remote service/infrastructure composed by the Markov model, PCFG and Neural Network-based probabilistic cracking tools. The configurations used for each cracking tool are detailed in Section 3.4.

The overall cracking results are depicted in Figure 4.3, where the total percentage of the number of cracked passwords is plotted as a function of the number of attempted guesses tried by each tool.

This figure shows that the Probabilistic Context Free Grammar (PCFG) tool cracked the largest number of passwords (almost 90%), while the other tools success rate ranged from 60% to 70% cracked passwords. Moreover, the PCFG and Markov Model tools needed fewer attempted guesses to reach a higher success rate, while the Neural Network-based probabilistic tool took many orders of magnitude higher in terms of number guesses, but still had a lower success rate than the former tools. As explained in Section 3.4, a significant part of the passwords in this sample are no longer than 8 characters, making them not properly suited for evaluation to this tool.

Moreover, Table 4.1 shows the number of passwords cracked under each dataset sample for this current experiment.

As expected, the number of cracked passwords under the unrelaxed conditions (detailed in Section 3.3) was higher, with the exception being for the Neural Network-based tool. A possible reason for

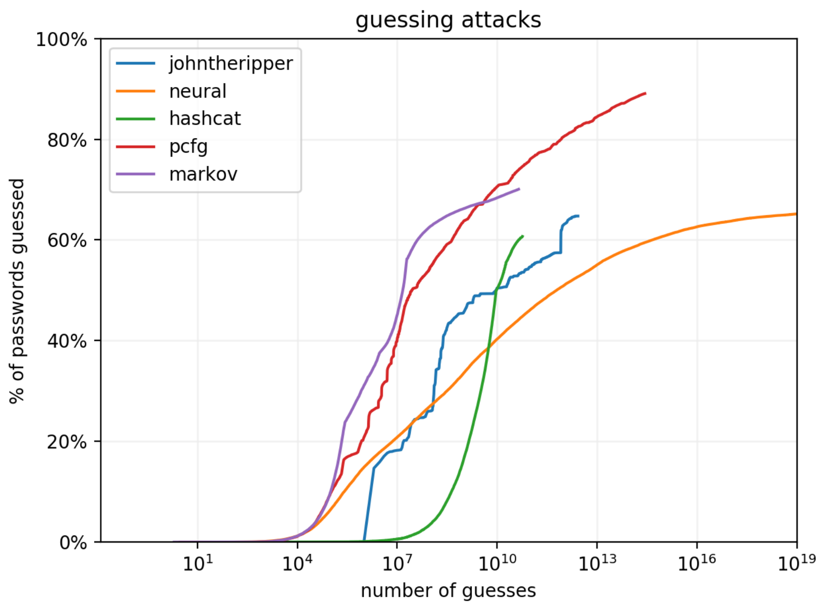


**Figure 4.3:** Password Guessing Resistance Results

| | *relaxed conditions* | | | *unrelaxed conditions* | | | |
|---|---|---|---|---|---|---|---|
| **Dataset Sample** | **RockYou** | **LinkedIn** | **000WebHost** | **RockYou** | **LinkedIn** | **000WebHost** | **Total (out of 60k passwords)** |
| JohnTheRipper | 5.5k | 4.8k | 2.4k | 9.6k | 9.2k | 7.4k | 38.9k (65%) |
| Hashcat | 4.7k | 3.7k | 2.1k | 9.6k | 9.1k | 7.2k | 36.4k (61%) |
| Markov Model | 9.9k | 3.7k | 1.9k | 10k | 9.5k | 7k | 42k (70%) |
| PCFG | 9.7k | 8.4k | 6.3k | 10k | 9.8k | 9.3k | 53.5k (89%) |
| Neural Network | 6.7k | 7.7k | 8.9k | 4.1k | 5.8k | 7.8k | 41k (68%) |

**Table 4.1:** Password Guessing Resistance by Dataset Sample

this, might be because these passwords are more frequent in leaked datasets and are, therefore, used as low-hanging fruits in the wordlists and training data of password guessing tools.

Finally, the JohnTheRipper and Hashcat cracking sessions took approximately 3 days and 1 hour, respectively, to complete. Even though Hashcat cracked fewer passwords than the JohnTheRipper cracking tool (under the already mentioned configurations), the time discrepancy between cracking sessions and the total amount of attempted guesses demonstrates that it was more efficient at guessing passwords. However, although the JohnTheRipper cracking session took way longer to complete, it ended up guessing and cracking a greater number of passwords. Moreover, it also needed fewer attempted guesses to achieve the 50% success rate mark.

### 4.2.3 Password Classification and Guessing Results Combined

Finally, we relate the PSMs classifications with the percentage of passwords cracked. For each cracking tool, all classification bins from the initial meters' distribution are split into two: one corresponding to the cracked passwords and another one to the uncracked passwords under that particular bin. Due to space limitations, we focus on the best performing tool of each cracking approach: JtR and PCFG.

Figure 4.4 shows the percentage of cracked (dotted bars) and uncracked (clear bars) passwords relative to each strength meter classification distribution, after attacking them with JtR and PCFG cracking tools (the remaining cracking tool results will be included in the Appendix A.2).

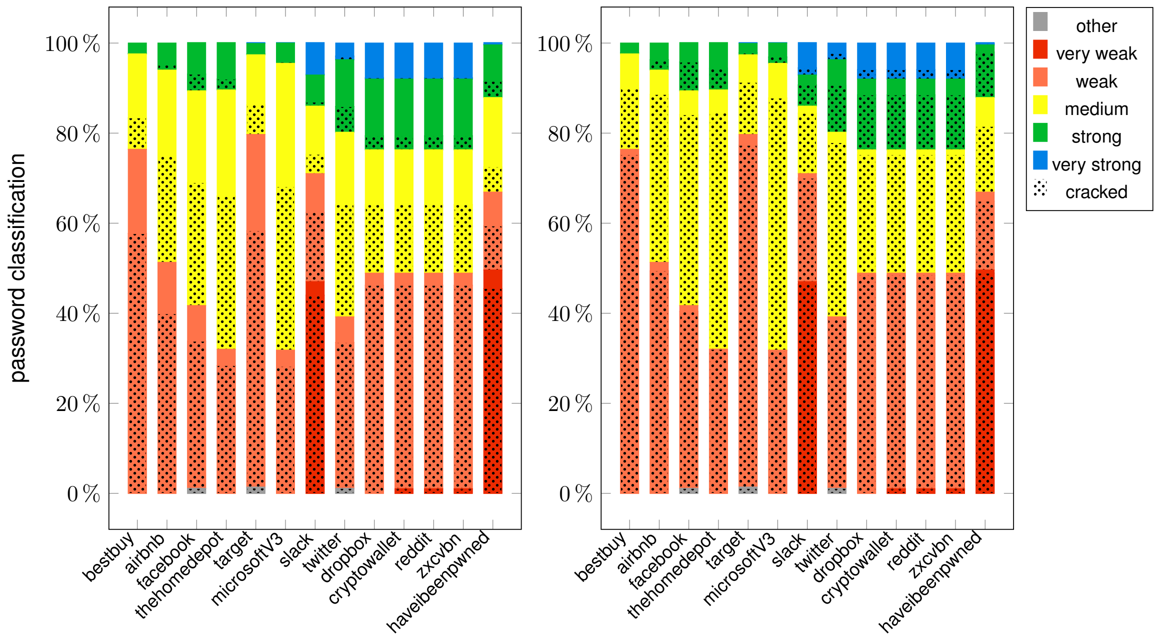


**Figure 4.4:** Cracked PSMs Classification Distributions with JtR (left) and PCFG (right) tools

When considering the JtR password cracking results, most passwords classified in the lower bins were cracked. Moreover, a little more than half and a very small part of the passwords classified in the middle and top bins were cracked, respectively. A similar pattern is observed when considering the PCFG tool (Figure 4.4, right), but the number of cracked passwords is considerably higher.

This confirms the expectation that passwords classified in the lower bins ("very weak", "weak" and "medium") are indeed easy to guess, whereas passwords classified in the stronger bins ("strong", "very strong") are harder to guess. This suggests that services should only accept passwords classified as strong and very strong. Nevertheless, both JtR and the PCFG tools cracked passwords in the stronger bins, suggesting that all meters can (and should) improve their password estimation methods.

When considering each dataset with respect to both cracking tools (Figures 4.5 and 4.6), we observe that the passwords from the 000WebHost dataset sample were harder to crack than the ones from the LinkedIn and RockYou. The LinkedIn password dataset sample was also slightly harder to crack when compared to the RockYou dataset sample.

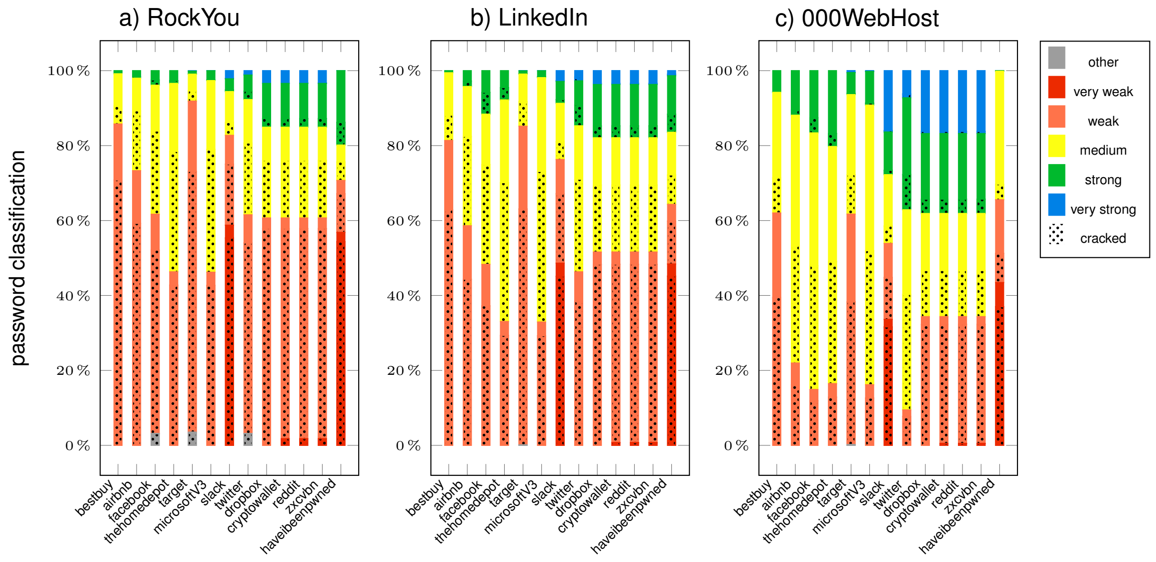


**Figure 4.5:** PSMs Classification Distributions of All Datasets According to JtR

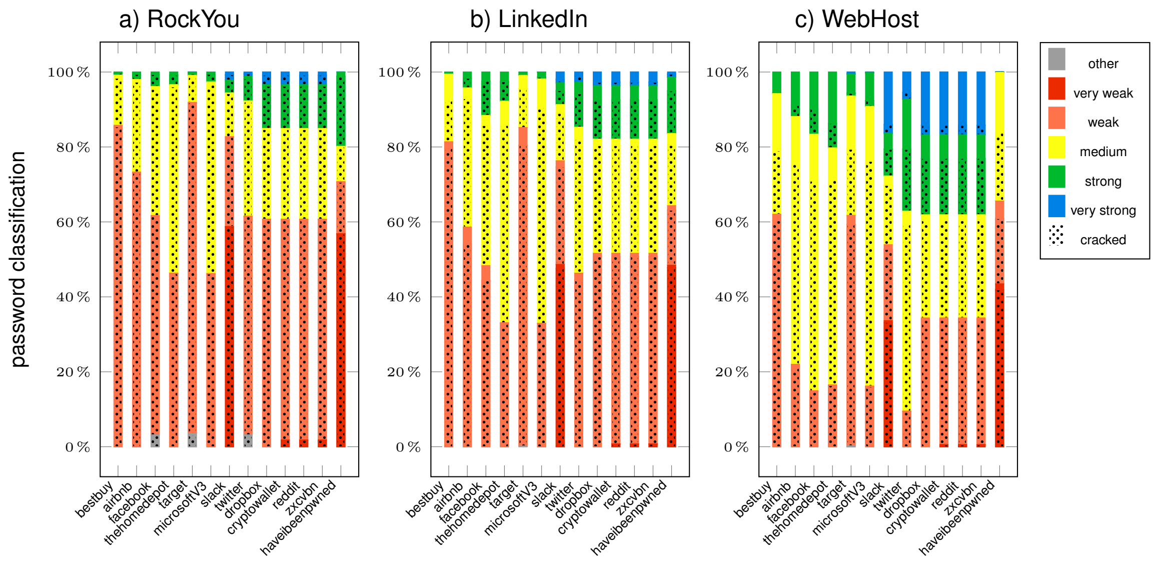


**Figure 4.6:** PSMs Classification Distributions of All Datasets According to PCFG

## 4.3 Discussion

This final section addresses the discussion of results regarding the initially proposed research question. Here we focus on: strength estimation, meter's accuracy and insights for improving these password security mechanisms in terms of both guessing resistance and accuracy.

### 4.3.1 Strength Estimation

Regarding password strength estimation between meters, our results show that some meters are considerably more conservative than others (Figure 4.1). We found that the most conservative are the Bestbuy and Target meters, with more than 96% of passwords classified with a maximum strength of medium and around 80% classified as weak. The less conservative are the Twitter and zxcvbn meters, with around 20% of passwords classified as strong or very strong.

Moreover, when considering the dataset samples individually (Figure 4.2), all meters, except haveibeenpwned, consider the 000WebHost passwords stronger than those in the other two datasets. Moreover, the LinkedIn password samples were classified as being stronger when compared to RockYou. This is likely due to the use of more stringent password composition policies under which the passwords contained in 000WebHost (lowercase and digits required and length$\geq$6) and LinkedIn (length$\geq$6) were created [2, 40].

The fact that haveibeenpwned meter does not consider the 000WebHost passwords stronger than those in the other two datasets suggests that passwords from 000WebHost are more frequent in the service. In fact, in our sample there were no passwords considered very strong by haveibeenpwned and only 0.1% were considered strong. Better quantization heuristics can be used in order to use this service as a strength meter.

### 4.3.2 Meter's Accuracy

Overall, we observe that passwords classified in the lower bins are more easily cracked than passwords classified in the upper bins (Figure 4.4). This suggests that password guessing resistance to off-the-shelf attacks of similarly labelled passwords relate to their password strength estimated by meters.

Looking at Figure 4.4, haveibeenpwned appears to be an exception with the ratio of cracked/uncracked passwords being greater in the strong bin than on the medium bin. However, this is justified by the fact that haveibeenpwned only considers 0.1% of 000WebHost passwords as strong and a substantial number of passwords classified as medium were not cracked, likely due to the more stringent composition policy (Figures 4.5 and 4.6). This also suggests that combining samples from different datasets that use different composition policies might lead to spurious results. It is thus important to undertake separate analysis on each sample.

Finally, the results show that only a small percentage of passwords classified as strong/very strong by zxcvbn have been cracked. This suggests that zxcvbn might be the best PSM in terms of accuracy and security. Nevertheless, a significant percentage of passwords classified as strong were cracked by the aforementioned cracking tools, suggesting that password strength estimation can be improved.

Moreover, since Reddit, Cryptowallet and Dropbox online services take advantage of zxcvbn as its

PSM, improving it in terms of accuracy might also have a positive impact on these services.

### 4.3.3 Insights

Based on the results obtained, we highlight the following points:

1. We advise that service providers should only accept passwords classified as strong or very strong, since the number of cracked passwords classified as medium or below is high;
2. The number of 000WebHost passwords cracked is much smaller than in other datasets (Figure 4.6c), suggesting that the use of more stringent password composition policies is advised;
3. Although our samples are built from leaked password datasets, some of the passwords contained in these do not appear as leaked in the service HaveIBeenPwned. Examples include the passwords “westsidetavern” (LinkedIn), “ozzyismybabe&ilovehimtodeath” (RockYou), and “asdfghjklqwertyuiopzxcvbnm1234” (000WebHost). Services that use HaveIBeenPwned in their password security checks should take this into consideration.

Finally, and by bridging our results with those of the CCS’18 work [2], we confirmed the weighted Spearman correlation sensitivity to the effects of quantization. According to our results, although some meters have almost identical password classification distributions (such as the zxcvbn and Dropbox or the Facebook and The Home Depot meters), they are classified very differently according to this metric (“ok” vs “very bad” and “ok” vs “bad”). This shows that small variations of the same password distribution can lead to a considerate discrepancy between results according to this metric, making it not tolerant to quantization effects.

# 5 PCP-based Analysis (RQ2)

**Contents**

This chapter explores our second research question, which is on the relation between the password guessing resistance and the evaluation of composition policies by strength meters.

As in the previous chapter, we will first outline the objectives regarding this research question and then describe the experimental designs that were conducted in order to answer it. Next, we present the obtained experimental results and, finally, at the end of this chapter we discuss the main takeaway insights derived from these preceding results and relate them with prior literature work.

## 5.1 Objectives and Experimental Design

The second research question that we set out to answer was:

> **RQ2**: Which PCPs are the most vulnerable/resilient against off-the-shelf guessing attacks? And how are they related to PSMs?

This question is concerned with investigating which PCPs are the most and least reliable against off-the-shelf password guessing attacks, and thus, fit for deployment in high-security contexts. Moreover, we want to analyze how passwords filtered according to different composition policies are evaluated by strength meters in order to establish a relationship between these two password security mechanisms.

As stated previously, PCPs have been deployed and embedded within strength meters to restrict the space of user-created passwords. The combination of composition/structural predictable patterns and semantic words/expressions within passwords decreases their overall guessing resistance. Thus, the aim of composition policies is to mitigate these factors, by forcing users to incorporate additional complexity into passwords in order to make them harder to guess, and hence, more secure against password guessing attacks.

The most common applications for composition policies in real-world online services are the explicit password creation policies, where users know upfront what are the rules and requirements needed in order to create acceptable passwords.

Examples of these rules include requiring a minimum length, usage of different character classes (LUDS - Lower, Upper, Digit, Symbol), blacklisting common words found in a dictionary, and avoiding structural patterns (such as l33t speak, keyboard walks or sequences).

However, under the influence of restrictive policies, users tend to have more difficulty in creating and recalling their passwords, leading to a greater struggle, frustration and self-induced/predictable coping strategies [32]. These usability behaviors potentially makes passwords more predictable, and thus more vulnerable against guessing attacks.

Moreover, there are also major differences between strength meters in terms of composition policy choices [29]. Many of them rely solely on minimum length requirements within their services, whereas others enforce a mixture of more complex policies.

Therefore, and despite depending on the security level to be achieved beforehand in a particular system, it is crucial to understand what are the impacts of selecting certain PCPs against guessing resistance and how they relate with strength meters and with human factors.

As such, our aim with this research question is to investigate the resilience of different composition policies against off-the-shelf password guessing attacks. Moreover, we want to understand how they relate to strength meters, in order to extract new insights from the obtained results and to build better password security mechanisms against guessing attacks.

To do this, we first determine how several composition policies withstand against widely popular guessing attacks in the password cracking community and in the academic literature in order to validate the results of previous password research. We then try to augment the gathered results by examining how these composition policies are evaluated by a wide selection of strength meters in order to establish a relationship between these two security mechanisms.

In order to achieve this objective and, thus, answer this research question we designed the following experiment:

- First, we randomly sampled 2,500 passwords from 14 different composition policies (enumerated in Section 3.2) for each previously filtered RockYou, LinkedIn and 000WebHost publicly available dataset leaks, having a grand total of 105,000 random passwords for this experiment;
- Second, and as was equally done in Chapter 4, we set out to attack these passwords by using two different guessing methods: exploiting local heuristic attack tools (JohnTheRipper and Hashcat) and taking advantage of a remote service/infrastructure (PGS by CMU) composed of probabilistic attack tools (Markov model, PCFG and Neural Network-based methods);
- We then set out to evaluate those 105,000 passwords by querying the 13 PSMs considered in this work. Each meter will therefore have its own password classification distribution (for each dataset sample) according to its own quantization scale per composition policy;
- Finally, by relating password guessing resistance with their respective meters' strength classifications, it is possible to analyze each meters' password distribution in light of cracked and uncracked passwords per bin. Moreover, we are able to perform a relative comparison between each composition policies' guessing resistance according to these very strength meter classifications.

**Experimental Setup.** All attack tools configurations are described in Section 3.4, while the local experiments were performed in a MacBook Pro laptop, running macOS Catalina (version 10.15.1) with the following specs: 2,7 GHz Intel Core i5 dual-core CPU, 8 GB RAM and Intel Iris Graphics 6100 1536 MB GPU.

## 5.2 Experimental Results

In this section we present the results of the experiments carried out in order to analyze the resilience of password composition policies against guessing attacks and how they relate to password strength meter's classifications.

### 5.2.1 Password Guessing Attack Results

First, we determined the guessing resistance of the 105,000 randomly sampled passwords through the use of two different cracking methods: heuristic and probabilistic cracking tools.

As in the previous experience presented in Chapter 4, we properly configured the JohnTheRipper and Hashcat heuristic cracking tools (both running on local hardware) and leveraged the CMUs' PGS remote service/infrastructure composed by the Markov model, PCFG and Neural Network-based probabilistic cracking tools. The configurations used for each cracking tool are detailed in Section 3.4.

The overall cracking results are depicted in Figure 5.1, where the total percentage of the number of cracked passwords is plotted as a function of the number of attempted guesses tried by each tool.

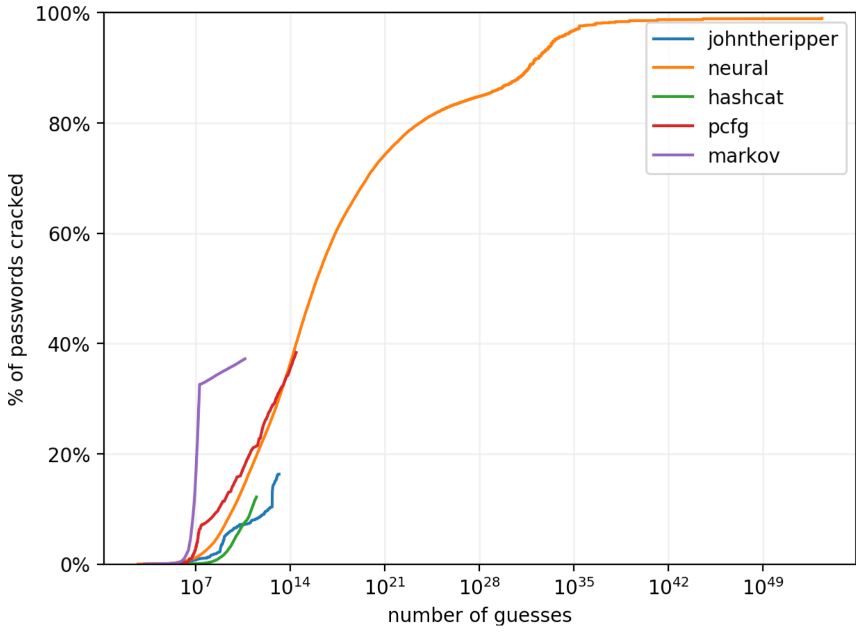


**Figure 5.1:** Password Guessing Resistance Results

As can be seen in the figure, it shows that the Neural Netword-based tool cracked the largest number of passwords by far. In fact, even though it took many orders of magnitude more in terms of guessing attempts in relation to the other cracking tools, it managed to crack almost 100% of the whole 105,000 password sample. This shows that an offline attacker using this cracking tool (or similar) in a high performance computation system could, in practice, compromise really hard to guess passwords even under stringent composition policies (such as 2word16 or 3class16).

As explained in Section 3.4 and in contrast with the cracking results achieved in the previous chapter, only a few number of passwords from this sample were not evaluated by the Neural Network-based probabilistic tool, because the great majority ranges from 8 to 31 character length.

Furthermore, both the Probabilistic Context Free Grammar (PCFG) and Markov model probabilistic cracking tools and the JohnTheRipper and Hashcat heuristic cracking tools had approximately the same success rate (around 38%,37% and 16%,12%, respectively), even though the PCFG and JohnTheRipper cracking tools computed a higher number of guessing attempts before exhaustion.

Moreover, Table 5.1 shows the number of passwords cracked under each dataset sample for this current experiment.

| **Dataset Sample** | **RockYou** | **LinkedIn** | **000WebHost** | **Total (out of 105k passwords)** |
|---|---|---|---|---|
| JohnTheRipper | 6.8k | 7.4k | 2.9k | 17.1k (16%) |
| Hashcat | 5.2k | 5.2k | 2.4k | 12.8k (12%) |
| Markov Model | 33.4k | 4.0k | 1.7k | 39.1k (37%) |
| PCFG | 20.7k | 13.1k | 6.4k | 40.2k (38%) |
| Neural Network | 34.4k | 34.7k | 34.8k | 103.9k (99%) |

**Table 5.1:** Password Guessing Resistance By Dataset Sample

As expected, the number of cracked passwords under the 000WebHost dataset sample were lower when compared to the RockYou and LinkedIn dataset samples, with the exception being for the Neural Network-based tool. This confirms the observation made in the previous chapter regarding the relative password strength for these publicly available datasets.

Finally, the JohnTheRipper and Hashcat cracking sessions took approximately 2 weeks and 4 hours, respectively, to complete. Just as with the experiment performed in the previous Chapter, although the JohnTheRipper cracking session took way longer to complete it also ended up computing more guessing attempts and cracking a greater number of passwords.

### 5.2.2 Composition Policy Ranking Results

Here we present the password composition policy rankings according to the previously gathered guessing attack results. As explained earlier, we randomly sampled 2,500 passwords from each previously filtered RockYou, LinkedIn and 000WebHost publicly available dataset leaks according to 14 different composition policies regularly studied in past password research works, such as in Shay's work [24].

These policy rankings are useful, because they give us information about how vulnerable/resilient each password composition policy is, when compared with each other, against heuristic and probabilistic cracking tools and thus, which policies should be applied in high-security services where guessing resistance should be seriously taken into account.

Figures 5.2 and 5.3 show the composition policy rankings according to the LinkedIn and 000WebHost datasets of 35,000 randomly sampled passwords. Each composition policy is ranked independently by to each cracking tool according to the number of cracked passwords under that same policy when compared to the others. Lower ranks mean higher guessing resistance against the aforementioned cracking tools used in this work.

Moreover, we decided to rule out the Neural Netword-based cracking tool due to its 99% success rate at cracking passwords. This means that all composition policies were equally ineffective against this particular cracking tool, thus limiting the PCP comparison and posterior analysis.

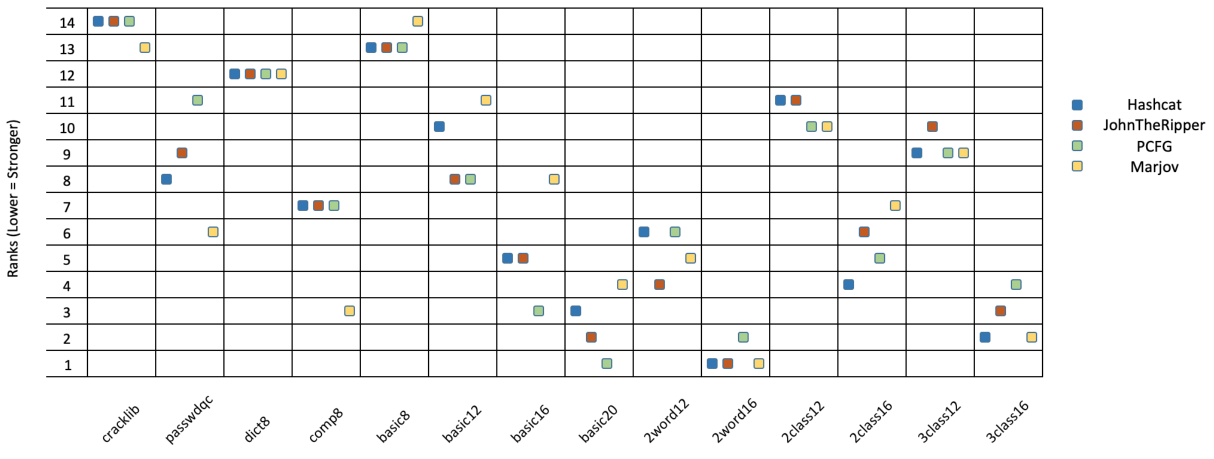


**Figure 5.2:** PCP Rankings of the LinkedIn Dataset Sample

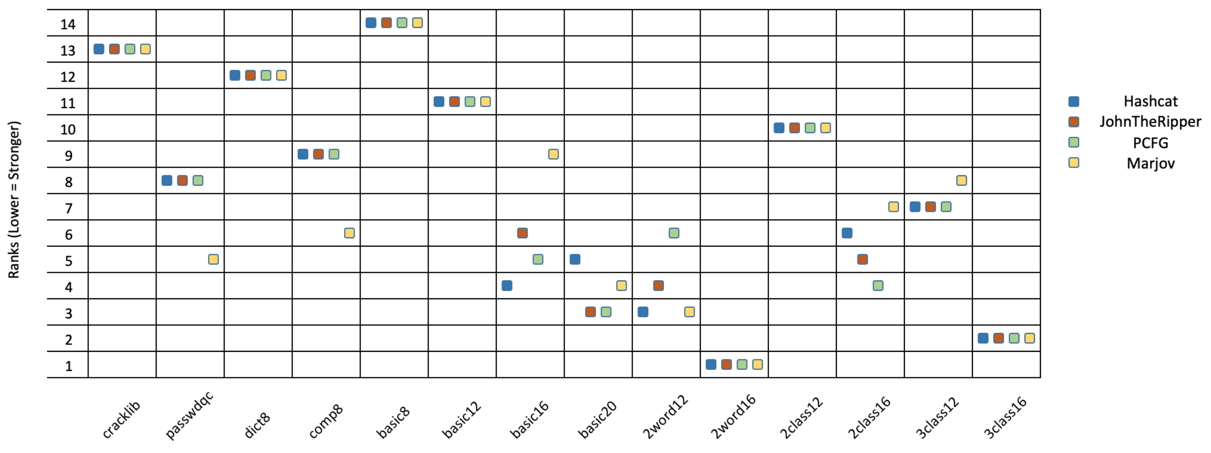


**Figure 5.3:** PCP Rankings of the 000WebHost Dataset Sample

Despite small policy ranking variations between each cracking tool, possibly derived from their different success rates, all policy rankings converge to the same results. As can be seen in both figures, we can divide these results into 3 distinct guessing policy ranking groups: weaker policies, composed by PAM_cracklib, basic8, basic12, dict8 and 2class12; stronger policies, composed by 2word16, 3class16 and basic20; and the remaining policies in between.

We also validate Shay's recommendations included in his study where password strength is tested under several permutations of length and character-based requirements [24]. More specifically, our results show that comprehensiveness combined with longer-length requirements (such as 2word16 and 3class16 policies) led to fewer easily guessed passwords than policies purely based on length (basic8, basic12 and basic 16), while still being more secure than a comprehensive policy with shorter length requirements (comp8).

These results suggest that more complex PCPs (with special emphasis on length plus character class requirements) provide a way for breaking users' predictable password selection habits for longer-length only requirements, thus making them more resistant against offline guessing attacks. On the other hand, simple policies (such as basic8 or dict8) and the PAM modules (that are widely deployed in Linux systems) should be avoided as they reveal little guessing resistance.

Furthermore, the policy rankings for the 000WebHost password dataset were more unanimous between each cracking tool when compared to the ones from the LinkedIn and RockYou datasets.

This unanimity might be explained due to the smaller discrepancy between the overall smaller number of cracked passwords under each cracking tool when compared with the other two datasets. Even though this discrepancy between cracking tools is somewhat more pronounced in the RockYou dataset sample (depicted in the Appendix A.3), the results still converge to the same underlying policy ranking groups previously identified.

### 5.2.3 Password Classification and Guessing Results Combined

Finally, we relate the PSMs classifications with the percentage of passwords cracked. For each cracking tool, all classification bins from the initial meters' distribution are split into two: one corresponding to the cracked passwords and another one to the uncracked passwords under that particular bin.

Due to space limitations, we focus our attention on the results produced by the PCFG probabilistic cracking tool under the RockYou dataset sample. The reason is that the total number of cracked passwords under this dataset was higher and therefore, more representative, when compared to the other two datasets. Moreover, after the Neural Network-based (which was ruled out for the reasons already mentioned), the PCFG was the best performing tool with 38% of success rate.

Figure 5.4 shows the percentage of cracked (dotted bars) and uncracked (clear bars) passwords relative to each strength meter classification distribution of the 35,000 RockYou dataset random sample, after attacking them with the PCFG cracking tool (the remaining cracking tool results are included in the Appendix A.3).

As can be seen it the figure, a great number of passwords are classified in the top bins. When comparing these password meter distributions with those from Figure 4.6 (left side), regarding the cracked

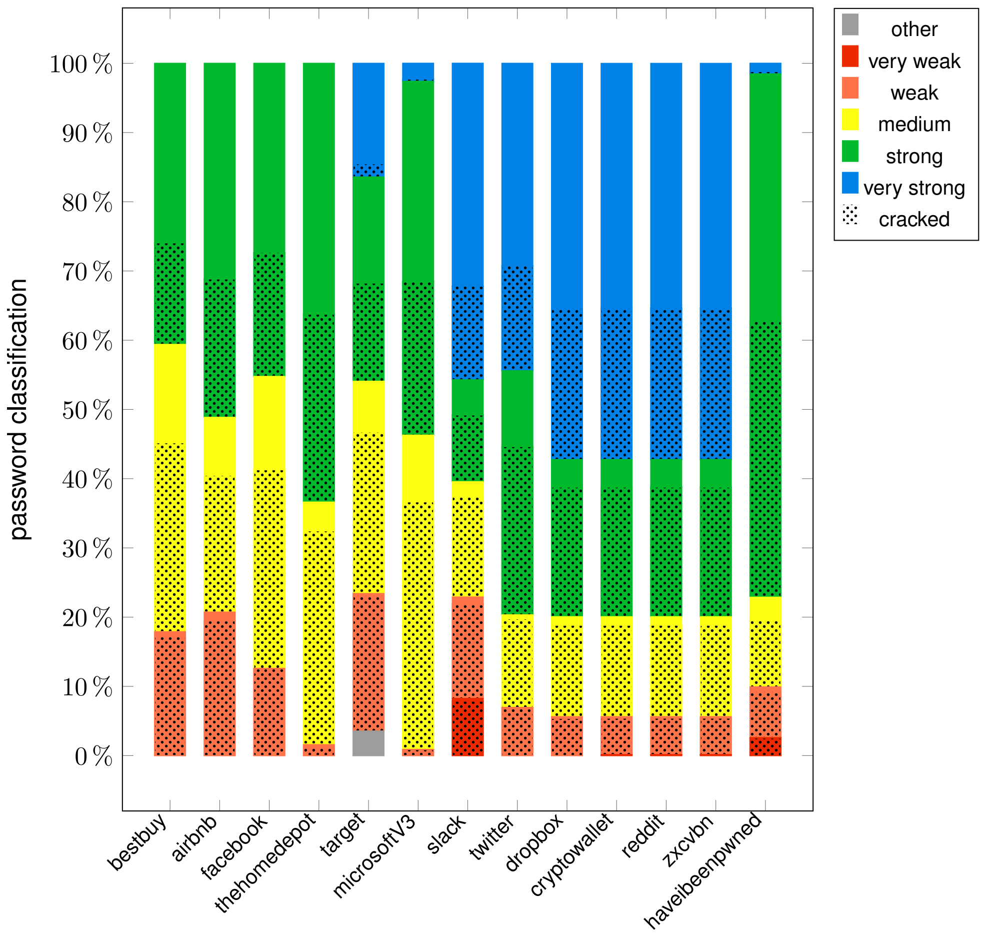


**Figure 5.4:** Cracked PSMs Classification Distributions with PCFG tool

passwords from the RockYou dataset under the PCFG tool of the previous chapter, we observe that filtering of passwords under composition policies yielded better overall scoring between meters.

Moreover, when considering the PCFG password cracking results, most passwords classified in the lower bins were cracked just like in the previous experiment from Chapter 4.

Once again, this confirms our expectation that passwords classified in the lower bins ("very weak", "weak" and "medium") are indeed easier to guess, whereas passwords classified in the top bins ("strong", "very strong") are harder to guess. This also suggests that services should only accept passwords classified on the top bins, because they denote better guessing resistance. Nevertheless, the PCFG tool cracked a significant part of the passwords classified in both the middle and top bins, suggesting that strength estimation could (and should) be further improved in order to make these meters more accurate.

Finally, Figures 5.5 and 5.6 showcase the PSMs distribution classifications according to both the top-3 worst (basic8, dict8 and PAM_cracklib) and best (2word16, 3class16 and basic20) composition policies.

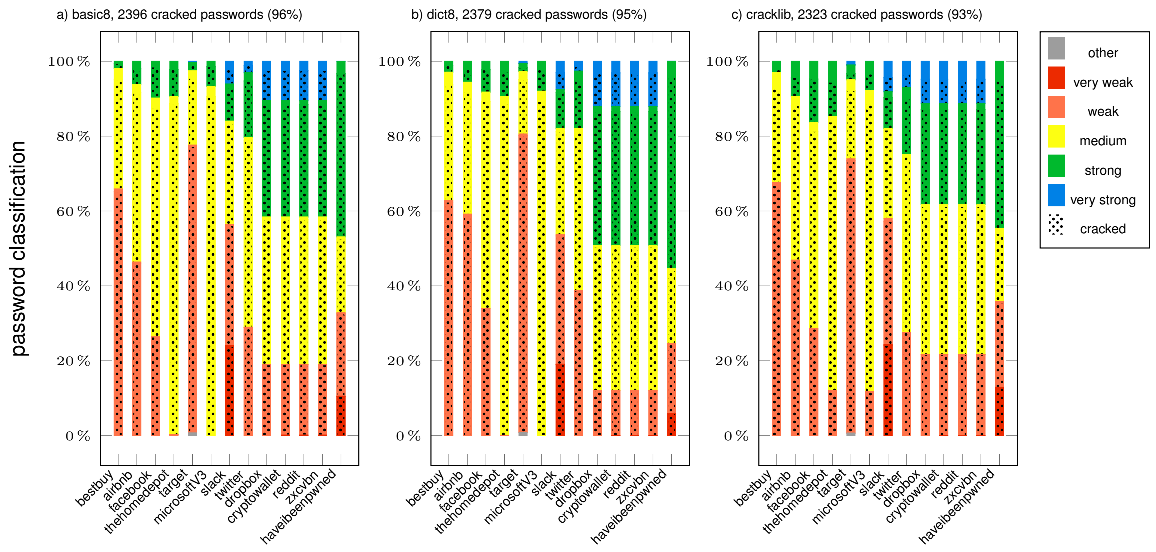


**Figure 5.5:** PSMs Classification Distributions for Top-3 Worst PCPs According to PCFG

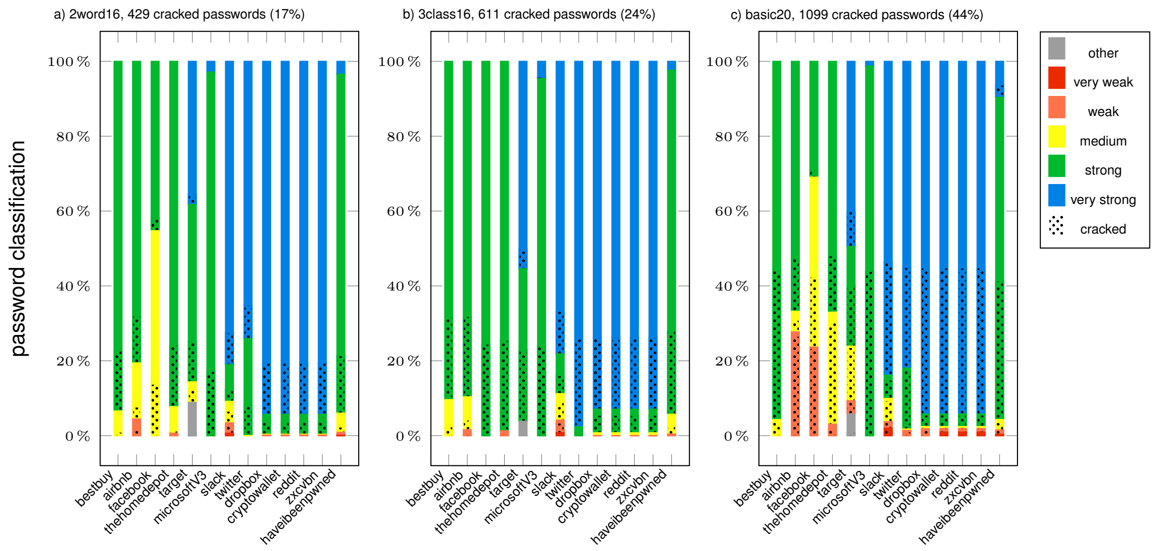


**Figure 5.6:** PSMs Classification Distributions for Top-3 Best PCPs According to PCFG

As can be seen in both figures, the way the top-3 worst and best composition policies are evaluated and portrait by PSMs differs completely.

On one hand, almost all passwords within to the basic8, dict8 and PAM_cracklib password policy samples have been cracked and more than half of these passwords were also classified in the lower bins by each meter. Moreover, even though there's still a small number of passwords classified in the top bins, these same passwords were nonetheless cracked, suggesting that, once again, the strength estimation methods employed in these meters could be further upgraded.

On the other hand, the great majority of the password policy samples from the top-3 composition policies (2word16, 3class16 and basic20) were classified in the top bins, with very few passwords being classified as being either "weak" or "medium". Even so, a non-negligible part of those passwords were still cracked, suggesting that some of these passwords were wrongly classified by meters. The remaining policy PSMs classification distributions are depicted in the Appendix A.3.

## 5.3 Discussion

This final section addresses the discussion of results regarding the initially proposed research question. Here we focus on: robustness of composition policies against guessing attacks and insights for improving password security mechanisms in terms of policy strength estimation and their usage within PSMs.

### 5.3.1 Robustness of Composition Policies

Regarding the robustness of password composition policies against offline guessing attacks, our results validate previous empirical research on this matter [24]. Not only did we show that composition policies such as basic8 and dict8, based on more relaxed requirements, are highly vulnerable against offline guessing attacks, thus making them not suitable to be incorporated within password security mechanisms, but also that composition policies such as 2word16 and 3class16, relying on a mixture of both longer-length and comprehensiveness requirements, rendered more resistant composition policies against these type of attacks (Figures 5.2 and 5.3).

These results suggest PCPs that place greater emphasis on length plus character class requirements helps breaking users' predictable password selection habits. However, given our attention solely on guessing resistance, we are unable to quantify the impact on usability and how much of a burden these policies represent to users.

In addition, we found that the PAM cracklib and passwdqc modules, which are used by default in widely deployed Linux systems, should be rendered unsuitable for utilization (Figure 5.2), because our results show that they are not reliable against offline guessing attacks.

Since many Linux distributions have long been used in critical systems, such as industry servers/devices, telecommunication equipment and other embedded systems, stricter policy modules (with tighter requirements) should be developed and shipped within these distributions.

### 5.3.2 Policy Strength Estimation

Regarding policy strength estimation, our results show that filtering passwords under composition policies yielded better overall scoring between meters (Figures 4.1 and 5.4).

Overall, we observe that composition policies belonging to the weakest ranking group are both classified in the lower bins by meters and fairly vulnerable against offline guessing attacks (since the number of cracked passwords classified as medium or below is high), whereas the ones belonging to the strongest ranking group are classified in the top bins and harder to crack (Figures 5.5 and 5.6). This suggests that services should not only reject passwords classified in the lower bins but also incorporate more complex composition policies into their PSMs, as they provide an explicit and safe way for filtering weak passwords beforehand.

Finally, even though 2word16, 3class16 and basic20 top-3 best composition policy samples in the experiment had most of its passwords classified in the top bins by each PSM, our results show that a non-significant percentage of these passwords were still cracked. This adds the following insight: only accepting passwords classified in the top bins and incorporating more complex composition policies into PSMs is not enough to further guarantee password safety against guessing attacks. Services should also focus their attention in improving the accuracy of their meters' strength estimation methods, especially in the strongest bins.

# 6

# Security Mechanism Enhancement (RQ3)

**Contents**

This chapter explores our third, and last, research question, which is on the extraction of new insights in order to build password security mechanisms with better strength estimation methods and accuracy. Due to our previous gathered results from the last two chapters, we decided to focus our attention on the zxcvbn meter as our targeted security mechanism to be improved.

We will first outline the objectives regarding this research question and then describe how the data acquisition was performed in order to tackle it. Next, we present how the zxcvbn's internal strength estimation methods work. Finally, we then identify some underlying issues regarding the accuracy of this meter and suggest adjustments that could address them.

## 6.1 Objectives

The third, and last, research question that we set out to answer was:

> **RQ3:** Is it possible to extract new insights from the obtained results in order to build password security mechanisms with better strength estimation and accuracy?

This question is concerned with interconnecting the previously gathered results with current password security mechanisms, by both analyzing and improving their strength estimation methods towards better accuracy and protecting password-based authentication against weak password selection and possible offline guessing attacks.

The experiments performed in Chapters 4 and 5 enabled us to gather a lot of useful data regarding the behavior of each PSM studied in this work. Not only did we gain information on how meters rate passwords in a general sense, but also which passwords are well and badly evaluated according to their guessing resistance.

More specifically, it is possible to individually analyze each PSM by cross-referencing their meter classification distribution with their associated guessing results in order to, for instance, identify passwords that were both cracked and classified in the stronger bins. That said, we can leverage this data for the purpose of finding issues regarding their strength estimation methods and then suggest further improvements to them.

We decided to focus our attention on the zxcvbn meter as our targeted security mechanism since, in addition to being an open-source[1] strength meter backed up by scientific work [28], our results showed that a significant percentage of passwords classified as strong by this meter were cracked by the cracking tools described in Section 3.4. This suggests that password strength estimation of the zxcvbn meter can be further improved. Moreover, since Reddit, Cryptowallet and Dropbox online services take advantage of zxcvbn as its PSM, improving it in terms of accuracy might also have a positive impact on these and many other services who might be also using it.

[1] GitHub repository: urlhttps://github.com/dropbox/zxcvbn

As such, our aim with this research question is to further identify inconsistencies on the zxcvbn's password classification results and then investigate their source regarding its internal strength estimation methods. Our intention is to further improve zxcvbn's accuracy without compromising its current features/architecture.

To do this, we first detail how the zxcvbn internal strength estimation methods work, so as to facilitate further analysis of our results. After this, we leverage the previously gathered results and filter them in relation to their guessing resistance, in order to identify passwords that were both cracked and evaluated as strong/very strong. Finally, after pinpointing these inconsistencies and their associated issues, we suggest further improvements that could be made to the zxcvbn meter in order to upgrade its accuracy.

## 6.2 zxcvbn Internal Architecture

As previously stated, the zxcvbn strength meter is an excellent candidate for further improvements. Our gathered results demonstrated both its overall good accuracy and the potential to make it even more accurate, specially passwords classified in the top bins. Moreover, this PSM has several properties that facilitate its extension: besides from being open-sourced, it is also small, fast, and easily portable (being crucially no harder to adopt than a LUDS approach [28]).

It consists of 3 sequential phases: match, estimation and search. Given an input plaintext password, it first models that password as consisting of one or more concatenated pattern matches. Next, during the estimation phase, each pattern is assigned an heuristic guess attempt estimation independently. Finally, the final phase searches for the sequence of adjacent matches that fully covers the input password while minimizing a total guess attempt figure.

In more detail, zxcvbn's internal structure is decomposed in the following phases:

- **Matching Phase:** enumerates all the (possibly overlapping) patterns it can detect. Currently zxcvbn matches: against several internal dictionaries (such as Wikipedia, English words, names and surnames, common passwords), reversed words, uppercasing and common l33t speak substitutions, repeated tokens ("aaaa"), spatial keyboard patterns ("qwerty"), alphabetic sequences ("12345", "abcdef"), date formats ("3-13-1997", "13.3.1997", "1331997") and brute-force patterns.

- **Estimation Phase:** a guess estimate is determined and assigned for each matched pattern. Assuming the attacker knows the patterns that make up a password, each guess estimate is heuristically determined based on how many guesses an attacker needs in order to guess that particular instance. For example: given a dictionary word, with a previously assigned $g$ guess estimates, repeated $n$ times, the number of guess attempts is estimated as being $g \cdot n$.

- **Search Phase:** given the full set of possible overlapping matches, zxcvbn finds the non-overlapping sequence of matched patterns that fully covers the whole input password while minimizing its number of guess attempts. For example, considering the password "notebook", zxcvbn would match 3 dictionary patterns: "note", "book", and "notebook" (assuming all these words are contained within its internal dictionaries). It would then assign guess estimates to each of those dictionary pattern matches individually. zxcvbn would then return the guess estimate for the complete word match, because, while covering the same password, computing the concatenation of the compound dictionary patterns would amount to more guesses being performed by a guessing attacker.

## 6.3 Inconsistencies and Possible Improvements

In this last section we present some of the major inconsistencies that we found during result analysis and posterior code inspection centered on the zxcvbn's strength estimation methods. The following summary contains some relevant observations/findings related to those inconsistencies as well as some possible fixes, which could be taken into account in order to further improve its accuracy at password strength estimation.

The data displayed below was gathered through:

- The gathered results of our previous experiments from Chapters 4 and 5 and posterior analysis. We focus our attention on passwords that were both cracked and classified as either "3 / 4" or "4 / 4" (these being the two highest bins) by the zxcvbn meter;
- The code inspection of **matching.py** and **scoring.py** files of the zxcvbn codebase at `https://github.com/dwolfhub/zxcvbn-python`, which is a Python ported library (recommended by the original developers);
- The HaveIBeenPwned service, which collects database dumps with information about billions of leaked accounts and their respective passwords. Here we are interested in the ranks (or equivalently, the frequency) for each password in every leaked dataset collected by this service.

### 6.3.1 At Matching Phase

We identified the following issues and possible adjustments to the matching.py file of the zxcvbn meter:

**1) Repeat Matching Function.** This function searches for repeated blocks of one or more characters within the password.

However, it only recognizes adjacent repeated blocks (such as "abcabc"). Even more, the presence of a single character between repeated blocks (such as "abc1abc") is sufficient to prevent this matching function to not recognize repeated but separated blocks, and subsequently to assign it its corresponding repeat guess estimate heuristic. The discrepancy between the assigned strength estimation of adjacent and non-adjacent repeated blocks is rather high and could be better modeled. More examples are shown in Table 6.1, with their respective HaveIBeenPwned frequency ranks.

| Non-Adjacent Blocks | zxcvbn | Rank | Adjacent Blocks | zxcvbn | Rank |
|---|---|---|---|---|---|
| 22Networking22 | 8.38385, 3 / 4 | 4 | Networking2222 | 5.98082, 1 / 4 | not found |
| asd123dsa123 | 8.33051, 3 / 4 | 70 | asddsa123123 | 5.58263, 1 / 4 | 54 |
| 1guiness1 | 8.00737, 3 / 4 | 45 | 11guiness | 5.39076, 1 / 4 | 2 |
| ***baby*** | 8.01062, 3 / 4 | 10 | baby****** | 4.5284, 1 / 4 | 4 |
| arrow1arrow2 | 9.37055, 3 / 4 | 35 | arrowarrow12 | 5.86652, 1 / 4 | not found |
| 1234fb1234 | 8.00004, 3 / 4 | 21 | 12341234fb | 4.30103, 1 / 4 | not found |

**Table 6.1:** Passwords With Repeated Blocks And Their Scoring

**Possible Improvements:** zxcvbn does not model inter-dependencies between pattern matches after the matching phase. This means that each pattern match is independently evaluated in the estimation phase without its associated context within the password itself. In order to solve this, we recommended the implementation of an extra layer in between the matching and estimation phase, capable of identifying non-adjacent repeated tokens (and other possible dependencies) of the same password. The guess attempt estimation heuristic for repeated blocks would then be applied correctly.

**2) Sequence Matching Function.** This function identifies sequences by looking for fixed Unicode codepoint differences between characters. For instance, the password “abcde” has a fixed Unicode codepoint difference of 1 between each character. However, it has two distinct problems.

Firstly, despite working fine for most regular sequences with a fixed Unicode codepoint difference, this matching function does not take into account sequences with Unicode codepoint differences regarding uppercase letters within it. A single example was found in our experiments (“Abcdefgh1”), but many more could be derived from it as presented in Table 6.2:

| Non-Sequence Patterns | zxcvbn | Rank | Sequence Patterns | zxcvbn | Rank |
|---|---|---|---|---|---|
| Abcdefgh1 | 8.00057, 3 / 4 | 508 | abcdefgh1 | 4.04532, 1 / 4 | 13 551 |
| ABCdefgh1 | 8.00004, 3 / 4 | 4 | | | |
| abCdEfgH1 | 9, 3 / 4 | not found | | | |

**Table 6.2:** Passwords With Non-Sequence Patterns And Their Scoring

Secondly, this matching function also does not take into account intercalated sequences, that is, sequences wrapped up in other sequences (“a9b8c7d6”) or non-sequence strings (“1r2r3r4r”). The discrepancy between assigned strength estimation of intercalated and non-intercalated sequence tokens is rather high and could be better modeled. More examples are shown in Table 6.3 with their respective HaveIBeenPwned frequency ranks.

**Possible Improvements:** lowercase all letters in the beginning of the sequence matching function, in order to properly identify the sequence pattern within the password, and then add the proper capital-

| Intercalated Sequences | zxcvbn | Rank | Non-Intercalated Sequences | zxcvbn | Rank |
|---|---|---|---|---|---|
| T1T2T3T4T5T6 | 12, 4 / 4 | 13 | 123456TTTTTT | 4.23553, 1 / 4 | not found |
| n1n2n3n4n5n6n7 | 14, 4 / 4 | 38 | 1234567nnnnnnn | 4.26482, 1 / 4 | not found |
| 0011223344556677 | 8.28869, 3 / 4 | 217 | 0123456701234567 | 1.82607, 0 / 4 | 12 |
| 010203040506070809 | 8.22775, 3 / 4 | 3180 | 000000000123456789 | 4.31806, 1 / 4 | 10 |
| k1k2k3k4k5k6 | 12, 4 / 4 | 373 | kkkkkk123456 | 4.23553, 1 / 4 | 59 |
| a1s2d3q4w5e6 | 9.73894, 3 / 4 | 140 | 123456asdqwe | 5.67247, 1 / 4 | 219 |

**Table 6.3:** Passwords With Intercalated Sequences And Their Scoring

ization heuristic bonus in the estimation phase. Moreover, this function could be extended in order to calculate the Unicode codepoint distance of both adjacent and non-adjacent characters (every two or even three characters for instance). Moreover, each identified sequence pattern match would then be evaluated independently in the estimation phase.

**3) Reverse Dictionary Matching Function.** This function reverses the input password and then calls the dictionary match function which searches for common passwords, English/wikipedia words and names and surnames, in its internal dictionaries.

However, this function does not recognize l33t speak substitutions if password is reversed (such as "n0itutitsbus"), thus matching it as a brute force pattern instead and assigning it a higher password strength estimation (as if it were recognized as a reverse pattern in the first place). This minor issue was identified through code inspection of the zxcvbn codebase. Therefore, no particular instances were found in our experiments, though some simple examples can be devised as presented in Table 6.4.

| Original | zxcvbn | Reversed | zxcvbn | Reversed + l33t | zxcvbn |
|---|---|---|---|---|---|
| computer | 1.716, 0 / 4 | retupmoc | 2.01284, 0 / 4 | retupm0c | 8, 2 / 4 |
| hackerman | 4.55609, 1 / 4 | namrekcah | 4.79214, 1 / 4 | n4mr3kc4h | 9, 3 / 4 |
| football | 1.17609, 0 / 4 | llabtoof | 1.4624, 1 / 4 | ll4bt00f | 8, 2 / 4 |
| iloveschool | 5.71105, 1 / 4 | loohcsevoli | 6.2515, 2 / 4 | l00hcs3v0l1 | 11, 4 / 4 |
| dictionary | 3.56074, 1 / 4 | yranoitcid | 3.86171, 1 / 4 | yranoitc1d | 10, 3 / 4 |

**Table 6.4:** Reversed Passwords With l33t Speak And Their Scoring

**Possible Fix:** this matching function should first call the l33t matching function and then reverse the password before calling the dictionary matching function in order to properly identify l33t speak substitutions and then correctly apply their corresponding heuristic strength estimations.

**4) Date Matching Function.** This function looks for dates recognized as any 3-tuple day-month-year mappings with 2 or 0 separator characters (such as "01-01-95" or "010195").

However it does not recognize dates with written-out months (such as "feb 31st") and odd delimiters other than "\, / -" (such as "01,01,95"). There were a significant number of cracked (English and non-English) passwords who exhibit such unidentified date patterns as can be seen in Table 6.5.

**Possible Fix:** convert (different language) month dictionary words into their corresponding numeral month date, in order to properly match the password as a date pattern. Moreover, current delimiters

| Original | zxcvbn | Equivalent Date | zxcvbn |
|---|---|---|---|
| 17april1989 | 8.00915, 3 / 4 | 17-04-1989 | 4.65572, 1 / 4 |
| 16thjune2007 | 8.25584, 3 / 4 | 16th062007 | 7.41858, 2 / 4 |
| 01october1989 | 8.01024, 3 / 4 | 01/08/1989 | 4.65572, 1 / 4 |
| 25,10,1992 | 8.00004, 3 / 4 | 25.10.1992 | 4.61152, 1 / 4 |

**Table 6.5:** Password Dates And Their Scoring

regex expressions should be extended to the date match function in order to recognize any repeated delimiter within the recognized mapping splits.

### 6.3.2 At Estimation Phase

As stated previously, during this phase a guess estimate is heuristically determined and assigned to each match. Despite this being enough to accurately and conservatively protect passwords within the range of an online guessing attack [28], our results show that current heuristic multipliers could be further extended in order to account for the matching patterns not currently identified by zxcvbn as well as the significant percentage cracked passwords that were wrongly classified as being strong or very strong by this meter.

Firstly, and as addressed by Johnson [43], passwords recognized as single tokens are inconsistently rewarded for capitalization. Matched dictionary tokens with non-letter characters (such as: "12345qwert") are not stripped before computing the capitalization heuristic multiplier. In this particular example, the meter awards the capital Q letter as if it were in the middle of the token (instead of its terminal position after stripping non-letter characters), thus granting it a higher multiplier score than predicted.

Furthermore, and as stated by the author of the zxcvbn work [28]: "*Unmatched regions are treated equally based on length(...) and unmatched digits and symbols are treated equally even though some are more common than other*". This means that the overall placement of non-lowercase letters within unmatched regions is not taken into account, as can be seen in the examples of Table 6.6.

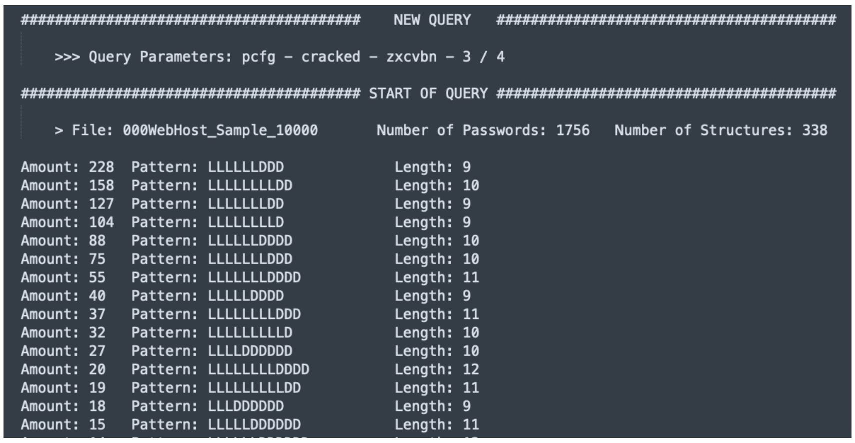
| Passwords | zxcvbn | Ranks |
|---|---|---|
| desenho | 7, 2 / 4 | 1100 |
| Desenho, desEnho, desenhO | 7, 2 / 4 | 4, not found, not found |
| desenho1, 1desenho, dese1nho | 8, 2 / 4 | 183, 9, not found |
| desenho!, des!enho, !desenho | 8, 2 / 4 | 8, not found, not found |

**Table 6.6:** zxcvbn Non-Lowercase Letter Placement Scoring

This issue might be even more problematic because common user-chosen pattern behaviors (such as using digit padding at the end of passwords, symbols as separators, uppercase letters at the start) being wrongly evaluated.

After analyzing the non-lowercase letter placement in our password datasets, we were able to confirm some of these user-chosen pattern behaviors. More specifically, from the RockYou, LinkedIn and 000WebHost datasets, respectively: 57%, 66% and 28% of passwords containing uppercase letters, end up placing them at the start; 57%, 63% and 61% of passwords containing digits, end up placing them at the end: and 48%, 51% and 52% of passwords containing symbols, end up placing them in the middle as a single separator.

Moreover, these patterns were also present in the cracked passwords classified as strong and very strong by zxcvbn from our results. Taking as an example the results from the experiment conducted in Chapter 4, when considering cracked passwords containing digits and classified as “3 / 4” by the zxcvbn meter from the 000WebHost relaxed password dataset sample against the PCFG cracking tool, our results showed that the top-15 most common cracked password structures (depicted in Figure 6.1) were all composed by a group of lowercase characters followed by a group of digits used as padding. This pattern is also repeated in other samples as well.



```
#######################################    NEW QUERY   #######################################

    >>> Query Parameters: pcfg - cracked - zxcvbn - 3 / 4

####################################### START OF QUERY #######################################

    > File: 000WebHost_Sample_10000       Number of Passwords: 1756   Number of Structures: 338

Amount: 228  Pattern: LLLLLLDDD           Length: 9
Amount: 158  Pattern: LLLLLLLLDD          Length: 10
Amount: 127  Pattern: LLLLLLLDD           Length: 9
Amount: 104  Pattern: LLLLLLLLD           Length: 9
Amount: 88   Pattern: LLLLLLDDDD          Length: 10
Amount: 75   Pattern: LLLLLLLDDD          Length: 10
Amount: 55   Pattern: LLLLLLLDDDD         Length: 11
Amount: 40   Pattern: LLLLLDDDD           Length: 9
Amount: 37   Pattern: LLLLLLLLDDD         Length: 11
Amount: 32   Pattern: LLLLLLLLLD          Length: 10
Amount: 27   Pattern: LLLLDDDDDD          Length: 10
Amount: 20   Pattern: LLLLLLLLDDDD        Length: 12
Amount: 19   Pattern: LLLLLLLLLDD         Length: 11
Amount: 18   Pattern: LLLDDDDDD           Length: 9
Amount: 15   Pattern: LLLLLDDDDDD         Length: 11
```

**Figure 6.1:** Common Cracked Password Structures From Chapter 4

Examining this particular example with more detail, from the 338 different structures regarding these cracked passwords: 151 structures end with a group of digits, 53 start with a group of digits, 36 have a group of digits separating between 2 groups of lowercase letters and 25 structures have a group of digits separating between lowercase letters and symbols. That is, taking only into consideration the four most common cracked password structures from this sample, at least 78% of these structures have predictable patterns concerning digit placement therein. Further analysis could be done for uppercase letter and symbol placements within cracked passwords in other samples from our work.

This issue hints that the password partitioning method and the usage of heuristic multipliers are not enough in order to accurately evaluate passwords in the stronger bins as they should be extended in order to better evaluate overall non-lowercase letter placement in unmatched regions.

**Possible Fix:** add a new data-driven estimation mechanism for evaluating non-lowercase letter placements in unmatched regions, and thus complement the already existing estimation methods for matched regions. By leveraging current rich data from publicly available passwords datasets, we would recommend performing a mapping of the underlying distributions of non-lowercase letter password placements and in order to identify predictable patterns contained therein. From there, a penalization/bonus could be applied to any unmatched region in function of the number of occurrences of those common pattern in relation with the overall frequency of all non-lowercase letter placement patterns. This way, the zxcvbn internal strength estimation methods would accurately model the effects of user password selection behaviors, instead of relying solely on heuristic guesses.

**Accuracy Testing.** Finally, the accuracy impact produced by the actual implementation of the aforementioned set of improvements in the zxcvbn's internal strength estimation methods could be later tested and compared using both our methodology and the one formulated by Golla and Dürmuth in their CCS'18 work [2].

# 7 Conclusion

**Contents**

In this thesis we have addressed the main problem of analyzing strength estimation of password security mechanisms currently used in online services and academia. Previous password research focused exclusively on evaluating the impacts of PSMs and PCPs against offline-attacks in real-world scenarios is rather scarce [2,24,29]. Therefore, our motivation was to study the relationship between these mechanisms and password guessing resistance, while providing new feedback to service providers and extending supporting evidence on how to develop better password security mechanisms in today's digital world. For this matter we decomposed the main problem in three research questions (defined in Chapter 1) that we set out to explore.

To answer these research questions, we defined a precise methodology (further detailed in Chapter 3) in order to assess currently used password security mechanisms and to analyse and compare their results with previous research work. We made use of publicly available dataset leaks, which were evaluated and filtered according to different PSMs and PCPs and then matched against their respective guessing resistance through the use of off-the-shelf offline-guessing attacks in order to better assess their accuracy on strength estimation and overall effectiveness. The ultimate objective was to cross reference each filtered subset of passwords with the corresponding password guessing resistance in order to produce new results for evaluation, comparison and analysis.

By following our methodology and after conducting several experiments (described in Chapters 4, 5 and 6), we were able to gather important results that helped us pointing out a set of relevant insights about password security mechanisms. Furthermore, it provided us with answers regarding our initially defined research questions. Namely, we were able to show that guessing resistance to off-the-shelf offline guessing attacks of similarly labelled passwords relate to their password strength estimated by PSMs. We also validated previous research on the robustness of PCPs against this type of attack, while examining how these filtered passwords are evaluated by PSMs. Moreover, we were able to identify several issues regarding the accuracy of the zxcvbn meter and also suggested improvements to its internal password strength estimation methods.

We therefore conclude that the initial research questions were properly addressed and explored, as they provided a better understanding on the relationship between password guessing resistance and strength estimation of both PSMs and PCPs, while providing new evidence and insights that are relevant on how to develop better password security mechanisms in today's digital world.

Finally, we also shared a public library of different utility Python scripts that were developed and used throughout this thesis[1], in order to attest the reproducibility of our results and to allow future extension by the password security community.

[1]GitHub Repository: https://github.com/davidfbpereira/pws_repo

## 7.1 Ethical Considerations

The set of available password datasets used throughout this thesis were illicitly stolen from breaches of several online web services [9–11], which culminated in users' credentials being publicly leaked and then shared throughout the digital space. Thus, the usage of this sensitive information in this work also raised some ethical concerns.

Taking this into account, we obtained the publicly available datasets, but we ensured that our use of this data would not inflict any further harm on its victims by excluding from this research any personal identifiable information, such as username credentials or email accounts and by not redistributing them in the wild.

Furthermore, since this data is widely shared after disclosure, any attacker with bad intentions could also take advantage of it by knowing in advance which tools are more effective at cracking passwords or by upgrading their cracking tools with better configurations. Moreover, these guessing attack methods are already available on the Internet and ready to be used off-the-shelf.

Our results might be used to inform attackers on which password guessing tools are more effective. However, our objective is to produce relevant information on how to improve the robustness of PSMs and similar security tools in order to reduce the success of these offline guessing attacks and thus, to mitigate further harm against users' sensitive data.

## 7.2 Threats to Validity

Our findings are limited to predominantly English speaking users and their password selection behaviors, as revealed by the password leaks used in this work. For that matter, further analysis should be extended in order to include password leaks from other native languages in order to validate our findings and further extend supporting evidence.

Finally, due to the lack of time and computational resources, we were not able to explore many different local heuristic cracking tool configurations, such as the use of even longer wordlists or by distributing the guessing workload over several CPUs/GPUs. One would expect that a professional password cracking attacker would normally use dedicated hardware combined with optimized configurations for its guessing approach [17]. Moreover, we were not able to locally configure and run the probabilistic cracking tools mentioned in Section 3.4, so we had to rely exclusively on the configuration possibilities provided by PGS.

Finally, we might have errors in our code that can affect some of the results. To mitigate this, we extensively tested our code. Moreover, in the spirit of open science, we make all our data and code available so that other researchers can inspect and reproduce our work.

## 7.3 Future Work

The material presented in Chapter 4 was published in RSDA 2020 [30], the 5th IEEE International Workshop on Reliability and Security Data Analysis co-located with the 31th Annual IEEE International Symposium on Software Reliability Engineering (ISSRE 2020). As a first next step, we are producing an extended version with the material from Chapters 5 and 6 that will be submitted soon.

We also consider that future password security mechanism analysis, with regards to the application of our methodology, could be further augmented by extending the selection of PSMs and PCPs under study. As been previously said in Chapter 3, we only included online web service meters in this thesis, but there are other online and offline services that make use of PSMs, such as academic meters, password manager applications and operating systems. Moreover, we envisage that this methodology could be further extended in order to design and compare improved variants of PCPs not only with regards to guessing resistance, but also to determine their impacts on usability.

Finally, and taking into account the set of identified issues and their possible adjustments suggested in Chapter 6, an upgraded version of the zxcvbn open-source meter could be developed in order to improve password strength estimation methods. Furthermore, its accuracy impact produced by the actual implementation could be later tested and compared using both our methodology and the one formulated by Golla and Dürmuth [2].

# Bibliography

[1] D. Florêncio, C. Herley, and P. v. Oorschot, “An administrator’s guide to internet password research,” *Proceedings of the 28th Large Installation System Administration Conference (LISA14)*, pp. 35–52, 2014.

[2] M. Golla and M. Dürmuth, “On the accuracy of password strength meters,” *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS’18)*, pp. 1567–1582, 2018.

[3] C. Herley and P. v. Oorschot, “A research agenda acknowledging the persistence of passwords,” *Published in IEEE Security and Privacy Magazine, Volume 10 Issue 1, Jan.-Feb.*, pp. 28–36, 2012.

[4] J. Bonneau, “The science of guessing: Analyzing an anonymized corpus of 70 million passwords,” *In IEEE Symposium on Security and Privacy (SP’12)*, pp. 538–552, 2012.

[5] D. Malone and K. Maher, “Investigating the distribution of password choices,” *Proceedings of the 21st international conference on World Wide Web (WWW’12)*, pp. 301–310, 2012.

[6] A. Vance, “If your password is 123456, just make it hackme,” *The New York Times*, 2010. [Online]. Available: https://www.nytimes.com/2010/01/21/technology/21password.html

[7] D. Florencio, C. Herley, and P. v. Oorschot, “Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts,” *Proceedings of the 23rd USENIX Security Symposium*, pp. 575–590, 2014.

[8] B. Ur, J. Bees, S. M. Segreti, L. Bauer, N. Christin, and L. F. Cranor, “Do users’ perceptions of password security match reality?” *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI ’16)*, pp. 3748–3760, 2016.

[9] T. Brewster, “13 million passwords appear to have leaked from this free web host,” *Forbes*, 2015. [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2015/10/28/000webhost-database-leak/#2f6a0e496098

[10] R. Hackett, "Linkedin lost 167 million account credentials in data breach," *Fortune*, 2016. [Online]. Available: http://fortune.com/2016/05/18/linkedin-data-breach-email-password

[11] J. Leyden, "Rockyou hack reveals easy-to-crack passwords," *The Register*, 2010. [Online]. Available: https://www.theregister.co.uk/2010/01/21/lame_passwords_exposed_by_rockyou_hack/

[12] L. Franceschi-Bicchierai, "Hacker tries to sell 427 milllion stolen myspace passwords for $2,800," *Vice*, 2016. [Online]. Available: https://www.vice.com/en_us/article/pgkk8v/427-million-myspace-passwords-emails-data-breach

[13] C. Warzel and M. Zeitlin, "It gets worse: The newest sony data breach exposes thousands of passwords," *Buzzfeed News*, 2014. [Online]. Available: https://www.buzzfeednews.com/article/charliewarzel/it-gets-worse-the-newest-sony-data-breach-exposes-thousands

[14] D. Goodin, "Why passwords have never been weaker—and crackers have never been stronger," *Ars Technica*, 2012. [Online]. Available: https://arstechnica.com/information-technology/2012/08/passwords-under-assault/

[15] ——, "Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331"," *Ars Technica*, 2013. [Online]. Available: https://arstechnica.com/information-technology/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/

[16] P. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. López, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," *In IEEE Symposium on Security and Privacy (SP'12)*, p. 523–537, 2012.

[17] B. Ur, S. M. Segreti, L. Bauer, N. Christin, L. F. Cranor, S. Komanduri, D. Kurilova, M. L. Mazurek, W. Melicher, and R. Shay, "Measuring real-world accuracies and biases in modeling password guessability," *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15)*, pp. 463–481, 2015.

[18] E. Liu, A. Nakanishi, M. Golla, D. Cash, and B. Ur, "Reasoning analytically about password-cracking software," *IEEE Symposium on Security and Privacy (SP'19)*, pp. 1272–1289, 2019.

[19] J. Hong, "The state of phishing attacks," *Magazine Communications of the ACM, January 2012, Vol. 55 No. 1*, pp. 74–81, 2012.

[20] D. Sukhram and T. Hayajneh, "Keystroke logs: Are strong passwords enough?" *IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pp. 619–625, 2017.

[21] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," *30th IEEE Symposium on Security and Privacy*, pp. 391–405, 2009.

[22] M. Dürmuth, F. Angelstorf, C. Castelluccia, D. Perito, and A. Chaabane, "Omen: Faster password guessing using an ordered markov enumerator," *Engineering Secure Software and Systems (ESSoS 2015)*, pp. 119–132, 2015.

[23] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: measuring the effect of password-composition policies," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, pp. 2595–2604, 2011.

[24] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Designing password policies for strength and usability," *ACM Journal Transactions on Information and System Security (TISSEC), Volume 18 Issue 4, May*, p. Article No. 13, 2016.

[25] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," *Proceedings of the 17th ACM conference on Computer and communications security (CCS '10)*, pp. 162–175, 2010.

[26] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis," *Proceedings of the 2010 IEEE INFOCOM*, pp. 1–9, 2010.

[27] M. Bishop and D. V. Klein, "Improving system security via proactive password checking," *Computers and Security, Volume 14, Issue 3 (IFIP)*, pp. 233–249, 1995.

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

[29] X. d. Carné de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS '14)*, 2014.

[30] D. Pereira, J. F. Ferreira, and A. Mendes, "Evaluating the accuracy of password strength meters using off-the-shelf guessing attacks," in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Aug 2020.

[31] G. A. Fowler, "Password managers have a security flaw. but you should still use one." *The Washington Post*, 2019. [Online]. Available: https://www.washingtonpost.com/technology/2019/02/19/password-managers-have-security-flaw-you-should-still-use-one/

[32] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: User attitudes and behaviors," *Proceedings of the Sixth Symposium on Usable Privacy and Security (SOUPS '10)*, p. Article No. 2, 2010.

[33] T. Nguyen, "Total number of synapses in the adult human neocortex," *Undergraduate Journal of Mathematical Modeling: One + Two, Vol. 3, Issue 1, Article 14*, 2010.

[34] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," *Proceedings of the 25th USENIX Security Symposium (SEC'16)*, pp. 175–191, 2016.

[35] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley, "Does my password go up to eleven?: The impact of password meters on password selection," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, pp. 2379–2388, 2013.

[36] D. Wang, D. He, H. Cheng, and P. Wan, "fuzzypsm: A new password strength meter using fuzzy probabilistic context-free grammars," *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 595–606, 2016.

[37] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models," *19th Annual Network and Distributed System Security Symposium (NDSS'12)*, 2012.

[38] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "How does your password measure up? the effect of strength meters on password creation," *Proceedings of the 21st USENIX Security Symposium*, pp. 65–80, 2012.

[39] M. Burgess, "Check if your LinkedIn account was hacked," https://bit.ly/33qwps0, May 2016, (Accessed on 02/08/2020).

[40] S. Johnson, J. F. Ferreira, A. Mendes, and J. Cordry, "Lost in disclosure: On the inference of password composition policies," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct 2019.

[41] J. F. Ferreira, S. A. Johnson, A. Mendes, and P. J. Brooke, "Certified password quality - A case study using Coq and Linux pluggable authentication modules," in *Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings*, 2017, pp. 407–421.

[42] S. Johnson, J. F. Ferreira, A. Mendes, and J. Cordry, "Skeptic: Automatic, justified and privacy-preserving password composition policy selection," in *Proc. AsiaCCS*, 2020.

[43] S. A. Johnson, “Passwords recognized as single tokens inconsistently rewarded for capitalization - issue #232 - dropbox/zxcvbn,” https://github.com/dropbox/zxcvbn/issues/232, 6 2018, (Accessed on 06/21/2018).

# A Experimental Results

## A.1 Local Guessing Attack Commands

Here we showcase the commands used in our local experiments regarding the heuristic cracking tools.

**Listing A.1:** JohnTheRipper

```
$ ./john --format=plaintext --wordlist="word.lst" --rules:Advanced passwords.hash
```

**Listing A.2:** Hashcat

```
$ hashcat -a 0 -m 99999 passwords.hash word.lst -r rules/advanced.rule --outfile-format=10
```

The “-a” switch specifies the attack mode (in this case, using a wordlist and mangling rules attack in order to derive guesses). The “-m” switch specifies the hash type (in this case, plaintext passwords). Finally, the “–outfile-format=10” switch defines the output file format for the recovered/cracked passwords (in this case, “password:guesses”).

## A.2 Chapter 4

Here we showcase the PSMs Classification Distributions for the Hashcat, Markov Model and Neural Network-based cracking tools.

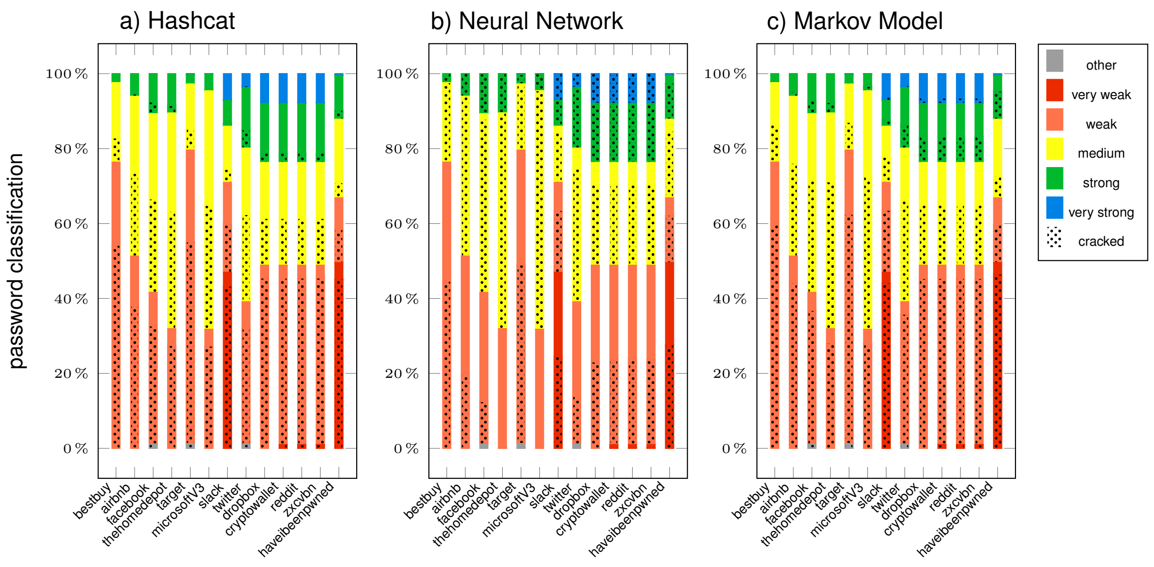


**Figure A.1:** Cracked PSMs Classification Distributions

## A.3 Chapter 5

Here we showcase: 1) the policy rankings for the RockYou dataset sample; 2) the PSMs Classification Distributions for the JohnTheRipper, Hashcat and Markov Model cracking tools; and 3) the PSMs Classification Distributions for the rest of the policy rankings according to the PCFG cracking tool.

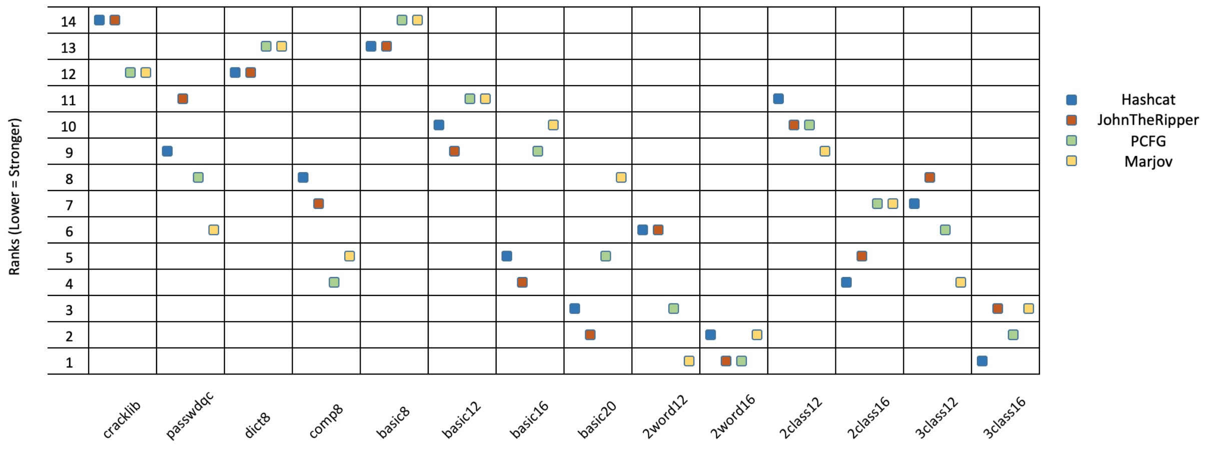


**Figure A.2:** PCP Rankings of the RockYou Dataset Sample

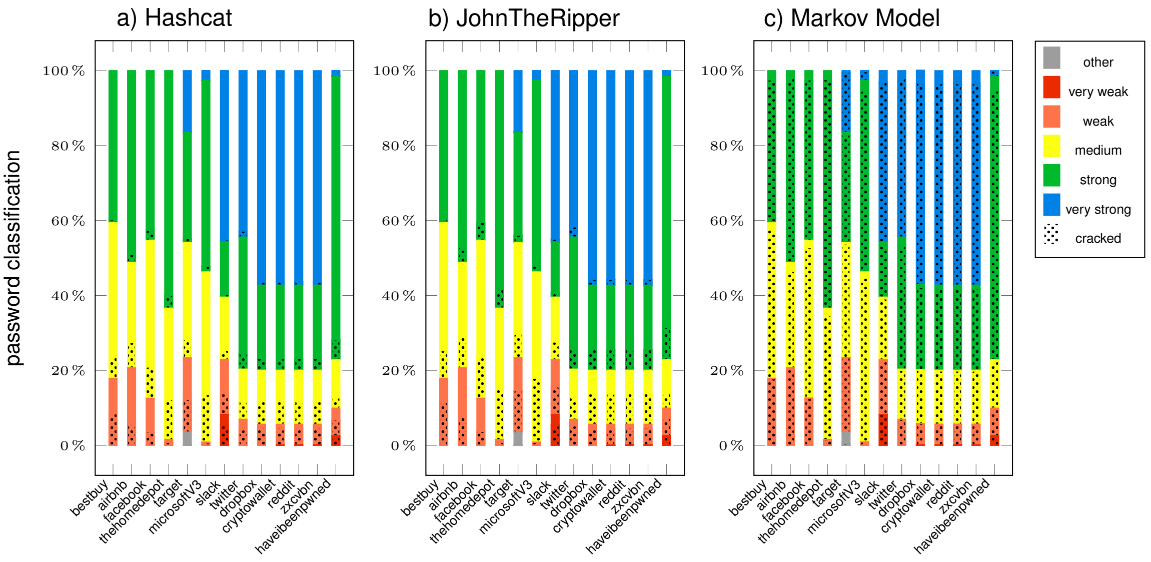


**Figure A.3:** Cracked PSMs Classification Distributions

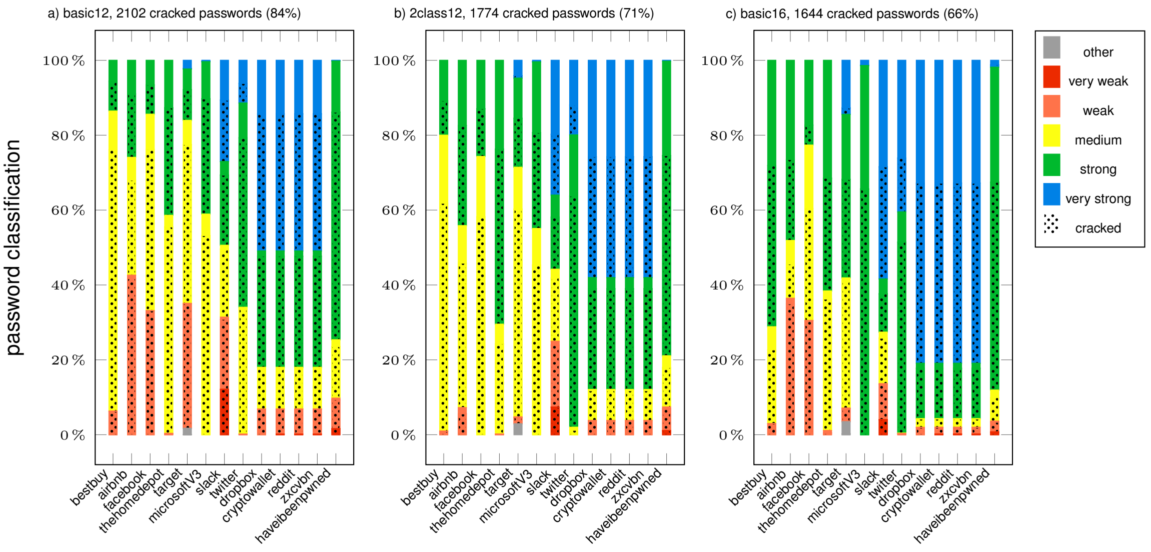


**Figure A.4:** basic12, 2class12, basic16 PSMs Classification Distributions According to PCFG

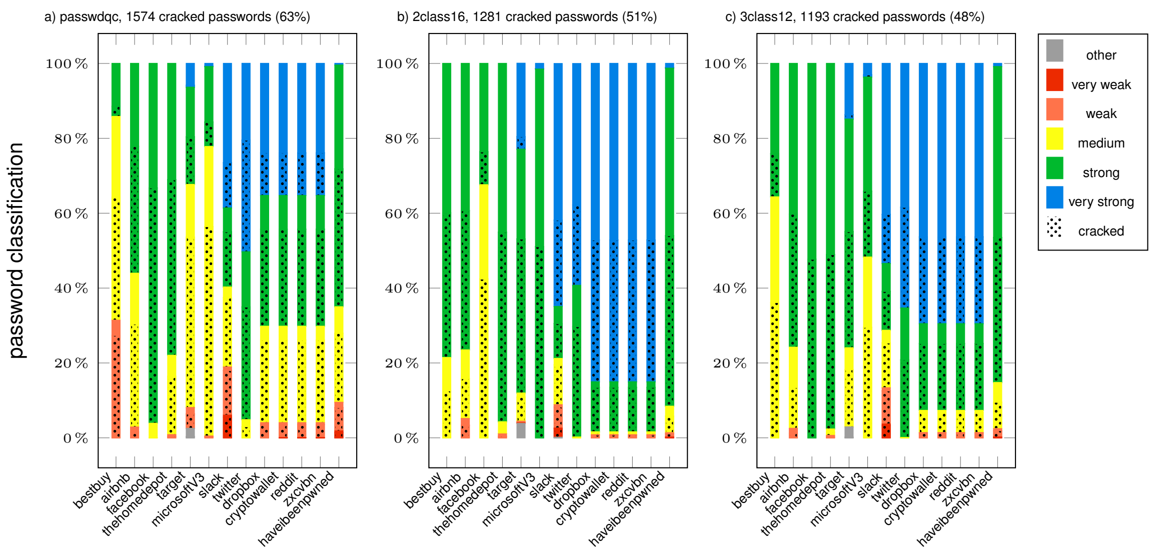


**Figure A.5:** passwdqc, 2class16, 3class12 PSMs Classification Distributions According to PCFG

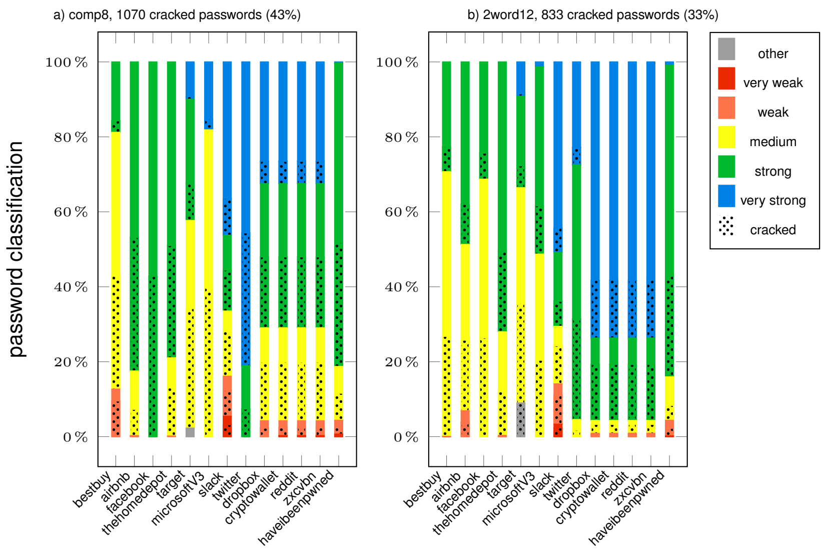


**Figure A.6:** comp8, 2word12 PSMs Classification Distributions According to PCFG