Evaluating Password Strength Meters and Password Composition Policies using Guessing Attacks

David Pereira*
INESC-ID & Instituto Superior Técnico
University of Lisbon, Lisbon, Portugal
Email: *david.b.pereira@tecnico.ulisboa.pt

***Abstract*—Passwords remain the primary authentication method in today's digital world. However, weak password selection behaviors combined with the re-utilization of credentials across services, make guessing attacks a serious threat against the integrity of user accounts. Password strength meters (PSMs) and password composition policies (PCPs) are security mechanisms that guide users towards better password selection. While recent studies have showed the efficacy of these security mechanisms, rigorous methods to assess their accuracy and to estimate password strength are needed.**

In this thesis we study the relationship between PSMs and PCPs with respect to their resistance against off-the-shelf guessing attacks by following a well-defined methodology. Our experimental results validate past password research and suggest new relevant findings regarding the security mechanisms under study. Finally, we focus on the zxcvbn meter by identifying several issues regarding its internal strength estimation mechanism and by proposing some adjustments so as to further improve its accuracy at password strength estimation.

***Index Terms*—Authentication; Password Strength Meter; Password Composition Policy; Accuracy; Guess Resistance**

I. Introduction

Passwords remain the primary authentication method in today's digital world and will likely prevail in the foreseeable future as a viable, practical and cheap method for user authentication [1], [2]. However, passwords alone do not guarantee the absence of security related issues in digital systems. Weak password selection behaviors [3], [4] combined with the re-utilization of credentials across different services [5], make guessing attacks a serious threat against the integrity of user accounts [6], [7]. In fact, the problem of maximizing resistance against password guessing attacks has been widely researched in academia for the past two decades [8], [6], [9], [10], [7].

Password strength meters (PSMs) and password composition policies (PCPs) are popular security mechanisms that help users choose stronger passwords. They rely on the idea of proactively checking password strength through estimation [11], [12], while enforcing certain requirements [5] and offering useful feedback to users.

Building an accurate PSM is one of the main challenges towards guiding users into better password selection. Depending on the metrics used for measuring password strength as well as how the passwords' estimation strength is computed, meters might misjudge, by over or underestimating, the true strength of passwords, thus failing to capture the passwords' guessing resistance [2], [6], [9]. This means that users trusting in inaccurate meters may actually be misguided into worse password selection.

Previous research focused exclusively on evaluating PSMs and PCPs is scarce [13], [14], [8], but adds crucial supporting evidence on how to develop better password security mechanisms and provide better feedback guidance towards a more secure user passwords selection in today's digital world. Therefore, our motivation is focused on better understanding the relationship between guessability and current password security mechanisms.

In this thesis, we study password guessing resistance against off-the-shelf guessing attacks as an accuracy and overall effectiveness measure of both PSMs and PCPs. We consider 13 PSMs, 14 PCPs, 5 different attack tools, and a random selection of 165,000 passwords extracted from three different datasets of real-world password leaks (RockYou [15], LinkedIn [16], and 000WebHost [17]).

We studied how passwords are evaluated and filtered according to different PSMs and PCPs and then relate them with their respective guessing resistance to guessing attacks. By following this methodology, we were able to: 1) validate past research and establish that guessing resistance against modern offline attacks can be used as an accuracy and effectiveness measure of PSMs and PCPs; 2) gather new supporting evidence and insights about which password security mechanisms are the most accurate at estimating passwords' strength and the ones that are the most vulnerable against guessing attacks; 3) we also identify a set of issues and possible improvements to the open-source zxcvbn meter in order to improve its accuracy; 4) finally, we share a public library composed by different utility Python scripts[1] developed in the context of this thesis, that could be leveraged or extended by the password security community.

After addressing the related work in Section II, we present the research questions and the evaluation methodology used in this study in Section III. In Sections IV, V and VI we present and discuss the obtained results. We conclude this work in Section VII, where we also discuss ethical considerations and future work.

II. Related Work

Research focused exclusively on evaluating PSMs is scarce. de Carné de Carnavalet and Mannan [14] analyzed 11 PSMs

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

deployed in popular websites by measuring the strength labels assigned to common passwords from several password dictionaries. They found evidence that the commonly used meters are highly inconsistent and fail to provide coherent feedback. Recently, Golla and Gürmuth [13] formulated a methodology for measuring the accuracy of a PSM. However, unlike the study presented here, none of these two approaches attempt to relate the output of PSMs with password guessing resistance to easily available, off-the-shelf guessing attacks.

New password cracking probabilistic methods [18], [19], [20] are being developed in academic research. 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 [21], [22], allowing billions of guesses per second.

The problem of maximizing password guessing resistance has been extensively researched [5], [6], [7]. A greater emphasis on studying password strength, on how to define and quantify it, has been carried out progressively [2], [6], [9], [13]. Moreover, new approaches with the aim of assisting and protecting users against modern password guessing attacks, through the development of effective security mechanisms [12], [19], have been introduced while trying to maintain the usability of passwords at the same time.

III. Study Design

This section presents the underlying evaluation methodology used in our experiments, including the research questions, the selection of the PSMs, PCPs, password datasets, attack tools, as well as the data collection and analysis methodology.

A. Research Questions

We aim to answer the following research questions:

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?

B. Password Strength Meters

We focus on PSMs that are used by popular web services and easily queryable, i.e. where the setup process together with password feeding and output scraping can be automated. Most of the PSMs 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. In addition, we include the *Have I Been Pwned?* service[3], which collects database dumps with information about billions of leaked accounts and their respective passwords.

[2]RankRanger Top 100 Websites: https://www.rankranger.com/top-websites

[3]Have I Been Pwned?: https://haveibeenpwned.com

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

- **zxcvbn** (5 bins) Popular academic PSM created by Daniel Wheeler [12]. We used the Python implementation.[4]
- **haveibeenpwned** This web service returns the frequency of a particular passwords' hash in the available leaked datasets.
- ***From popular websites:***
  - 3 bins: **airbnb**, airbnb.com; **bestbuy**, bestbuy.com; **thehomedepot**, homedepot.com
  - 4 bins: **dropbox**, dropbox.com; **target**, target.com; **facebook**, facebook.com; **microsoftV3**, bit.ly/39LCXT6
  - 5 bins: **cryptowallet**, blockchain.com; **reddit**, reddit.com; **slack**, slack.com; **twitter**, twitter.com

C. Password Composition Policies

We selected the following PCPs based from [8]. This set is composed by a mixture of length, character-class and dictionary requirements 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 the **cracklib** and **passwdqc** Linux Pluggable Authentication Modules (PAM), because they consist of software packages which enforce composition policies that are used by default in widely deployed Linux systems [23].

D. Password Datasets

The datasets of leaked passwords that we consider in this study are the following: 1) **RockYou**, compromised in plaintext from the *RockYou* online gaming service of the same name around the year 2009 [15]. The version we obtained contained 32,603,048 passwords; 2) **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 contained 15,271,208 passwords; 3) **LinkedIn**, compromised from the professional social networking site *LinkedIn* around the year 2012 [16]. Unsalted password hashes

[4]zxcvbn Python module: https://github.com/dwolfhub/zxcvbn-python

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. The version we obtained contained 172,428,238 passwords.

Data cleansing and filtering: As recommended in this type of studies [24], each dataset was first filtered according to the password composition policy it is known to have been created under [25]. Passwords containing non-ASCII characters were then removed to avoid encoding issues that might arise due to multi-byte characters being stored as multiple characters, artificially inflating password length. Finally, as shown by Bonneau [10], 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.

E. Attack Tools

To study password guessing resistance we selected two different conceptual approaches 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.[5] While JtR and Hashcat include wordlists and rule lists samples, they are far smaller than those used in typical attacks and far more ineffective [7]. Therefore we adopted an advanced configuration, by making use of wordlists far more extensive[6] (with near 304,000 and 1,600,000 common password entries and natural-language dictionaries). Moreover, we combined the stock (151), SpiderLabs (5,146) and Megatron (15,329) mangling rules for JtR and the Best64 (77), T0XlC (4,085) and Generated2 (65,117) mangling rules for Hashcat. The PGS probabilistic cracking tools were trained as detailed in the PGS website and we used their recommended configurations.

F. Data Collection and Analysis

In order to answer the proposed research questions, we: 1) randomly sampled 10,000 passwords from each previously filtered RockYou, LinkedIn and 000WebHost publicly leaked datasets (with both relaxed and unrelaxed frequency conditions), having a grand total of 60,000 random passwords for the first experiment; 2) randomly sampled 2,500 passwords from our PCP selection for each previously filtered RockYou, LinkedIn and 000WebHost publicly available datasets, having a grand total of 105,000 random passwords for the second experiment; 3) we queried the 13 password strength meters

[5]PGS service: https://pgs.ece.cmu.edu

[6]https://github.com/berzerk0/Probable-Wordlists

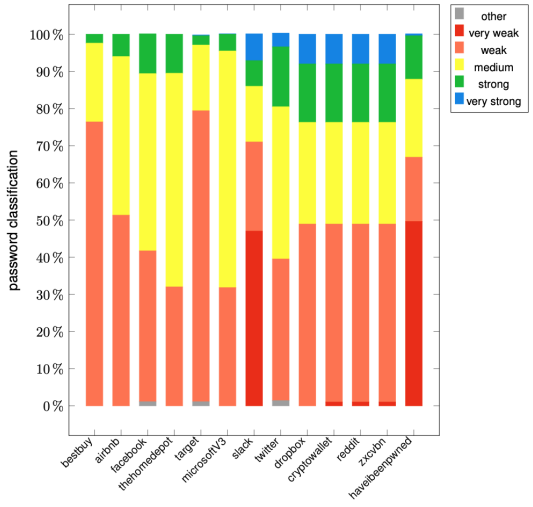

Fig. 1. PSMs Classification Distributions

considered in this study with those 165,000 passwords. Each meter produced its own password classification distribution (for each dataset sample) according to its own quantization scale; 4) we then set out to attack those passwords by using the attack tools described above; 5) finally, by relating password guessing resistance with their respective meters' strength classifications and filtered policy, we analyzed each meters' password distribution in light of cracked and uncracked passwords per bin.

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

IV. PSM-Based Analysis (RQ1)

This section presents the results of the experiment carried out in order to analyze the accuracy of PSMs.

A. Password Meter Classification Results

We start with the password classification distributions for each PSM. These 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. These results are useful because they give us information about the estimation behaviour of each PSM.

Figure 1 shows the password classification distributions for each PSM on the whole sample of 60,000 random passwords. Each percentage corresponds to the number of passwords under each meters' classification quantization bin.

Each meter has its own quantization scale which is represented by a different colour. The great majority of these bins are represented by a textual or numerical representation, 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

[7]All the code used is available: https://github.com/davidfbpereira/pws_repo

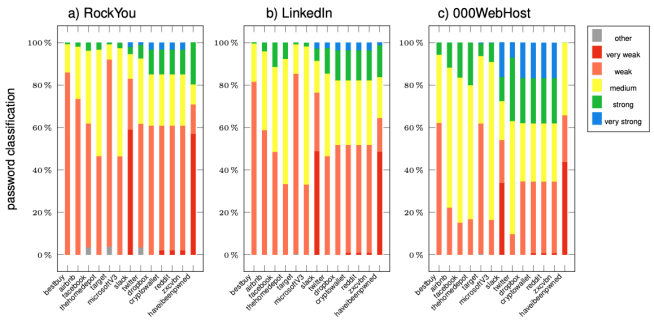

Fig. 2. PSMs Classification Distributions for all Datasets

the PSMs **twitter**, **facebook** and **target**) 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 **haveibeenpwned** outputs the number of occurrences for each password, we decided to map that number to one of 5 bins: a password not found was deemed "very strong", 1 occurrence was deemed "strong", 2 to 5 occurrences deemed "medium", 6 to 50 occurrences deemed "weak" and over 50 deemed "very weak".

The PSMs **dropbox**, **cryptowallet** and **reddit** produce almost the same password classification distribution results as **zxcvbn**(**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.

Finally, we can observe four distinct meter groups, namely: conservative meters in both the lower and higher bins (**bestbuy** and **target** services); less conservative meters in the lower bins but not the higher ones (**airbnb**, **facebook**, **thehomedepot** and **microsoftV3**); conservative meters in the lower bins but not the higher ones (**slack**); and less conservative meters in both lower and higher bins (**twitter**, **zxcvbn** and its derivatives).

The quantization scale used for **haveibeenpwned** 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 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.

B. Password Guessing Attack Results

The overall cracking results are depicted in Figure 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

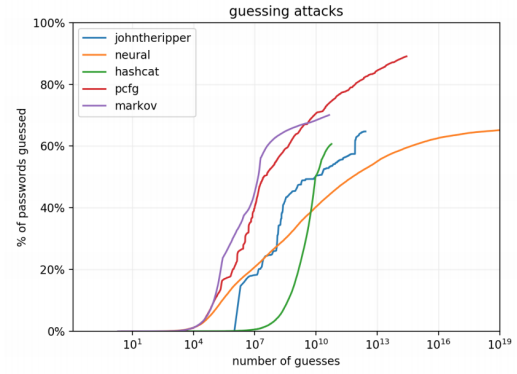

Fig. 3. Password Guessing Resistance Results

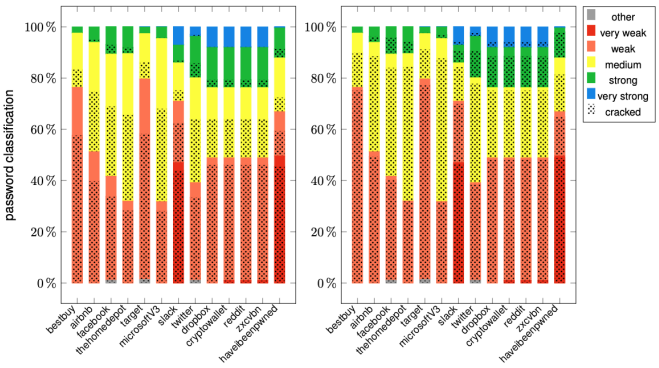

Fig. 4. Cracked PSMs Classification Distributions with JtR (left) and PCFG (right) tools

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; the neural network-based probabilistic tool took many orders of magnitude higher in terms of number of guesses, but still had a lower success rate than the former tools.

Table I 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 was higher, with the exception being for the neural network-based tool. A possible reason for 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.

C. Password Classification and Guessing Results Combined

Finally, we relate the PSMs classifications with the percentage of passwords cracked. Due to space limitations, we focus on the best performing tool of each cracking approach: JtR and PCFG.

Figure 4 shows the percentage of cracked (dotted bars) and uncracked (clear bars) passwords relative to each PSM classification.

When considering the JtR password guessing results, most passwords classified in the lowest 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,

TABLE I
PASSWORD GUESSING RESISTANCE BY DATASET SAMPLE

	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%)

respectively. A similar pattern is observed when considering the PCFG tool (Figure 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 and 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.

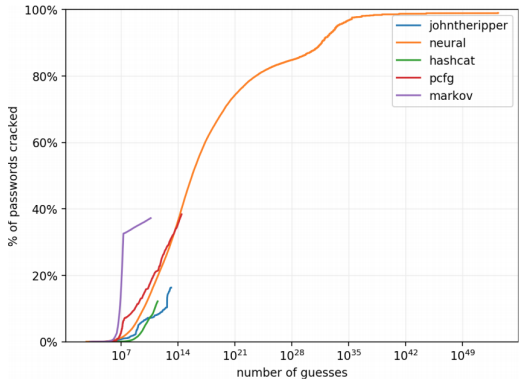

Fig. 5. Password Guessing Resistance Results

D. Discussion

Strength Estimation: Regarding password strength estimation between meters, our results show that some meters are considerably more conservative than others (Figure 1).

Moreover, when considering the dataset samples individually (Figure 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 [13], [25].

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). This suggests that password guessing resistance to off-the-shelf attacks of similarly labelled passwords relate to their password strength estimated by meters.

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.

Finally, and by bridging our results with those of the CCS'18 work [13], 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.

V. PCP-BASED ANALYSIS (RQ2)

This section presents the results of the experiment carried out in order to analyze the effectiveness of PCPs.

A. Password Guessing Attack Results

First, we determined the guessing resistance of the 105,000 randomly sampled passwords. The overall cracking results are depicted in Figure 5. It shows that the Neural Network-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. Furthermore, both the 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, Figure 6 shows the number of passwords cracked under each dataset sample for this current experiment.

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

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%)

Fig. 6. Password Guessing Resistance by Dataset Sample

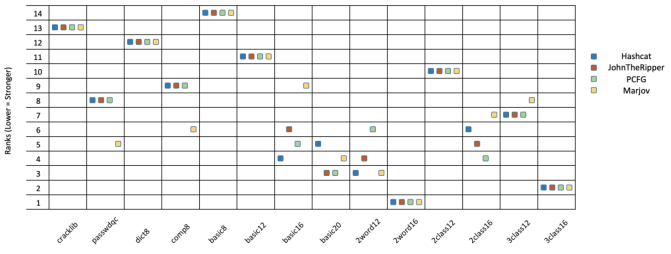

Fig. 7. Password Guessing Resistance by Dataset Sample

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.

B. Composition Policy Ranking Results

Here we present the password composition policy rankings according to the previously gathered guessing attack results. Figure 7 shows the composition policy rankings according to the 000WebHost dataset 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.

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 [8]. 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.

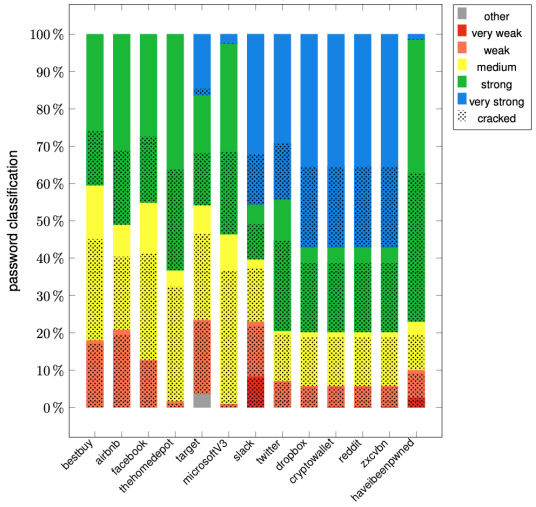

Fig. 8. Cracked PSMs Classification Distributions with PCFG tool

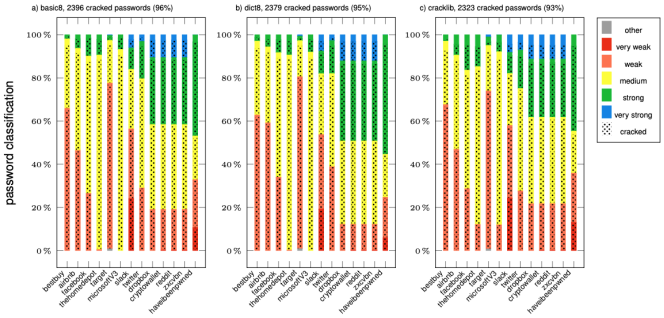

Fig. 9. PSMs Classification Distributions for Top-3 Worst PCPs According to PCFG

C. Password Classification and Guessing Results Combined

Finally, we relate the PSMs classifications with the percentage of passwords cracked. Due to space limitations, we focus our attention on the results produced by the PCFG probabilistic cracking tool under the RockYou dataset sample.

Figure 8 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. A great number of passwords are classified in the top bins. When comparing these password meter distributions with those from Figure 4 (left side), regarding the cracked 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.

Finally, Figures 9 and 10 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. 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

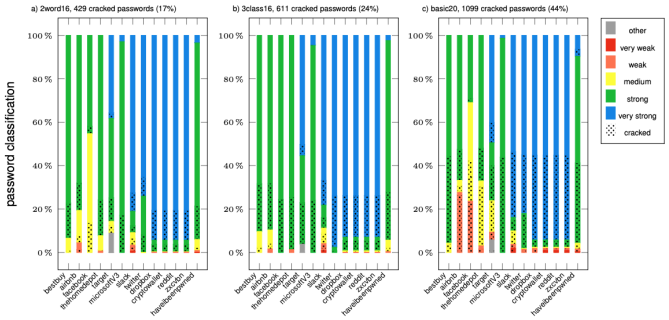

Fig. 10. PSMs Classification Distributions for Top-3 Best PCPs According to PCFG

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.

D. Discussion

Robustness of Composition Policies: Regarding the robustness of password composition policies against offline guessing attacks, our results validate previous empirical research on this matter [8]. 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, 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 (Figure 7). These results suggest PCPs that place greater emphasis on length plus character class requirements helps breaking users' predictable password selection habits.

In addition, we also found that the cracklib and passwdqc Linux modules, should be rendered unsuitable for utilization (Figure 7). 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.

Policy Strength Estimation: Regarding policy strength estimation, our results show that filtering passwords under composition policies yielded better overall scoring between meters (Figures 1 and 8).

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 9 and 10). 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.

VI. Security Mechanism Enhancement (RQ3)

The conducted experiments 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.

We decided to focus our attention on the zxcvbn meter as our targeted security mechanism since, in addition to being an open-source[8] strength meter backed up by scientific work [12], our results showed that a significant percentage of passwords classified as strong by this meter were cracked, suggesting that it could be further improved.

To do this, we first detail how the zxcvbn internal strength estimation methods work. 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.

A. zxcvbn Internal Architecture

zxcvbn's internal structure is composed by 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.

B. Inconsistencies and Possible Improvements

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

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

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.

Possible Improvements: zxcvbn does not model interdependencies 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 (such as “Abcdefgh1” found in our results).

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”).

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 capitalization 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).

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 (“feb 31st”) and odd delimiters other than “\, / -”. There were a significant number of cracked (English and non-English) passwords who exhibit such unidentified date patterns in our results.

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

5) Unmatched Scoring Function.

Firstly, and as addressed by Johnson [26], 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 [12]: “*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 II.

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 II
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 Section IV, 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

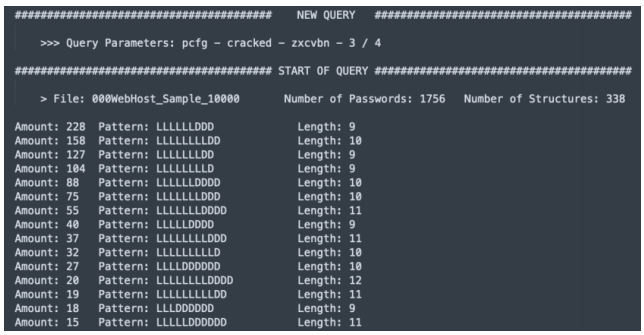

Fig. 11. Common Cracked Password Structures From Section IV

(depicted in Figure 11) 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.

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 [13].

VII. Conclusion

We have addressed the main problem of analyzing strength estimation of password security mechanisms currently used in online services and academia. Our motivation was to study the relationship between PSMs and PCPs 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.

We defined a precise methodology 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.

After conducting several experiments, we were able to gather 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, as they provided a better understanding on the relationship between password guessing resistance and strength estimation of both PSMs and PCPs, while providing new supporting evidence 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[9], in order to attest the reproducibility of our results and to allow future extension by the password security community.

A. Ethical Considerations

The set of available password datasets used throughout this thesis were illicitly stolen from breaches of several online web services [27], [28], [17], 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.

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

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.

B. Future Work

The material presented in Section IV was published in RSDA 2020 [29], 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 Sections V and VI 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 Section III, 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 Section VI, 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 [13].

REFERENCES

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

[2] D. Florêncio, C. Herley, and P. C. van Oorschot, "An administrator's guide to internet password research," in *Proc. LISA*, 2014.

[3] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proc. WWW*, 2012.

[4] A. Vance, "If your password is 123456, just make it hackme," *The New York Times*, 2010. [Online]. Available: https://nyti.ms/3guN6WH

[5] 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," in *Proc. CHI*, 2011.

[6] 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 *Proc. IEEE Symp. Security & Privacy*, 2012.

[7] 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," in *Proc. USENIX Security*, 2015.

[8] 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," in *ACM Journal Transactions on Information and System Security (TISSEC), Volume 18 Issue 4, May*. ACM, 2016, p. Article No. 13.

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

[10] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE Symp. Security & Privacy*, 2012.

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

[12] D. L. Wheeler, "zxcvbn: Low-budget password strength estimation," in *Proc. USENIX Security*, 2016.

[13] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *Proc. CCS*, 2018.

[14] X. de Carné de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Proc. NDSS*, 2014.

[15] N. Cubrilovich, "Rockyou hack: From bad to worse," https://tcrn.ch/2PoXZNW, Dec 2009, (Accessed on 02/08/2020).

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

[17] T. Brewster, "13 million passwords appear to have leaked from this free web host," *Forbes*, 2015, (Accessed on 02/08/2020). [Online]. Available: https://bit.ly/33saroy

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

[19] 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," in *Proc. USENIX Security*, 2016.

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

[21] D. Goodin, "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/

[22] ——, "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/

[23] 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 *Proc. Integrated Formal Methods*, 2017.

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

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

[26] 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).

[27] 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/

[28] 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

[29] 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.