PCWQ: A Framework for Evaluating Password Cracking Wordlist Quality

Aikaterini Kanta[1,2], Iwen Coisel[2], and Mark Scanlon[1(✉)]

[1] Forensics and Security Research Group, School of Computer Science, University College Dublin, Dublin, Ireland
aikaterini.kanta@ucdconnect.ie, mark.scanlon@ucd.ie
[2] European Commission Joint Research Centre (DG JRC), Via Enrico Fermi 2749, 21027 Ispra, Italy
{aikaterini.kanta,iwen.coisel}@ec.europa.eu
https://www.forensicsandsecurity.com/

Abstract. The persistence of the single password as a method of authentication has driven both the efforts of system administrators to nudge users to choose stronger, safer passwords and elevated the sophistication of the password cracking methods chosen by their adversaries. In this constantly moving landscape, the use of wordlists to create smarter password cracking candidates begs the question of whether there is a way to assess which is better. In this paper, we present a novel modular framework to measure the quality of input wordlists according to several interconnecting metrics. Furthermore, we have conducted a preliminary analysis where we assess different input wordlists to showcase the framework's evaluation process.

Keywords: Password cracking · Wordlist · Dictionary · Contextual information

1 Introduction

Despite known security concerns, passwords still remain the most widely used and one of the easiest and most adopted methods of authentication. As password policies become more restrictive by enforcing the selection of stronger passwords, attacks also become more refined and sophisticated. Traditional password cracking methods have, in many cases, become less efficient due to the increase of the computational cost of the underlying algorithms and the strengthening of the passwords [3]. Salting the passwords[1] also drastically increases the complexity of the password cracking process when targeting several passwords as each salt must be consider independently.

In the case where a single password is considered, if the attacker is aware of information regarding the target, that information can be leveraged aiming at

[1] The salt is a random string (typically 3 to 5 random characters) that is concatenated to the password before hashing it. Identical passwords therefore have a different hash.

P. Gladyshev et al. (Eds.): ICDF2C 2021, LNICST 441, pp. 159–175, 2022.
https://doi.org/10.1007/978-3-031-06365-7_10

cracking the password in fewer attempts. An example of this type of situation would be a law enforcement officer wanting to access a password protected digital device of a suspect during the course of an investigation. In this case, time is of the essence in order to swiftly resolve the investigation or prevent further criminal acts.

Another particular case is when the targeted dataset can be associated to a particular context. An example would be a penetration testing campaign evaluating the strength of the passwords used by the user of a system. If such system is linked to a particular community - for example users of a video game service - the operator can use contextual information such as typical language used in this community, references from the topic, or any other type of contextual information to refine the password cracking process.

All the above information can be useful, to try to make more educated guesses about the password of a target or a community [10]. The ultimate goal is to recover the password faster than we would with current state-of-the-art approaches, by giving them a head start regarding the wordlist they use during the password cracking process. By creating a custom wordlist, that is tailored to the target and by checking first password candidates that are more likely to be chosen as the password by the target, we can have a more efficient password recovery process.

In this article, we have designed a modular framework to assess the quality of a wordlist to be used in password cracking processes. We have proposed several criteria that can be considered for such evaluation and explained why there cannot be a single and totally ordered metric to evaluate the wordlist. Our methodology relies on the Password Guessing Framework (PGF)[2], initially designed to evaluate and compare password guessing tools, that we have adapted and complemented for the need of this study. We are then discussing how this framework can and will be used to evaluate and improve wordlist generation processes depending on the conditions of particular scenarios.

2 Background and Related Work

2.1 Password Cracking Techniques

There is a vast array of password cracking techniques, that are used depending the situation, from the traditional, like an exhaustive search, to the recently developed, like machine and deep learning techniques, such as the ones based on Generative Adversarial Networks (GANs) [7]. A common approach is the dictionary attack in combination with mangling rules, which are grammar and slang substitutions and modifications that aim to imitate human tendencies during password selection. For example, using the number *3* instead of the letter *e*, adding a *!* at the end of the password, capitalising the first letter, etc. It is important to note that mangling rules widen the set of guesses significantly, because each new alteration has to be checked with all the password candidates. This is

[2] https://www.password-guessing.org/.

why a balance has to be achieved between the number of mangling rules that we want to test and the time we can afford for the recovery process. There are common mangling rules that are used by the community such as the Hashcat[3] Best64 rules.

Other common password cracking methods include rainbow tables [15], which are based on the idea of a time-memory trade-off where the hashes of plaintext passwords are pre-computed and stored for faster lookup. A common countermeasure to rainbow tables is the use of a `salt`, which is a random value concatenated to the plaintext password before it is hashed, rendering rainbow tables unpractical in all scenarios where salt is used.

There are many different password cracking algorithms, some of which have been around for years, like John the Ripper[4] and hashcat[5], and some which have emerged lately such as PassGan [7] which uses a Generative Adversarial Network (GAN) to learn password rules directly from leaked password lists, or the neural network based solution from Melicher et al. [14]. Along with those already mentioned, several other tools have been proposed and we cite few of them in what follows. OMEN [4] is using a Markov model to generate candidates in a decreasing order of probabilities. PCFG [25] which stands for Probabilistic Context-Free grammar, where the input dictionary is used to create a context-free grammar and assign probabilities to it. PRINCE[6] creates intelligent chains to all combinations of words from the input wordlist.

2.2 Analysis of Password Trends

Something else that has changed in recent years is that information about previous passwords is easily available on the internet. Every day we hear of new data breaches that expose passwords (plaintext or hashed) and sometimes accompanying information, such as emails, names, addresses, etc. For an attacker wanting to access a specific account belonging to an individual, the password cracking attack could be reduced to a simple lookup for a match in leaked lists. In fact, studies of password habits of users have shown that users tend to reuse passwords that they need to enter frequently [24] and they tend to underestimate the consequences of doing so.

Furthermore, even when passwords are not reused explicitly, there are password ties between older and newer passwords of the same user [19]. But, even in the good case where that particular user does not reuse the same password across different services, knowing their previous passwords or other information about them, can give great insight to the cracking process [9]. To this end, TarGuess, a framework that makes use of Personally Identifiable Information (PII) and cross-site information, has been proposed to make targeted guesses of users' passwords and it was shown to outperform current models for both the cases of non-savvy and security-savvy internet users [23].

[3] https://hashcat.net/.
[4] https://www.openwall.com/john/.
[5] https://www.hashcat.net.
[6] https://github.com/hashcat/princeprocessor.

But even if previous passwords of a user are not known, a case can be made that knowledge of passwords of other users can speedup the process [11]. In fact, studies have shown that there are common misconceptions people fall prey to when creating their password, such as thinking that by adding a symbol at the end of the password they make it more secure [6,18].

The password policy in place plays a role at the strength and guessability of a password. It has been shown that putting a password policy in place forces users to have more secure passwords, whereas users left to their own devices will generally choose weaker passwords [13]. But the stronger passwords required by password policies may lead users to either having trouble remembering and ending up writing down their passwords [12] or to use common techniques for bypassing the requirements without building a strong password [16]. For example, `Password1!` fulfills the requirement for uppercase, lowercase, symbols and numbers and is above eight characters, but would not be considered a strong password.

2.3 Password Strength Meter

Password strength meter is another field of study that is continuously evolving following the sophistication of password cracking attacks. The most basic strength meter is a simple 0/1 metric where basic rules must be respected by the password to accept it, and reject it otherwise, like the LUDS-8 from the NIST proposal back in 2004 [1]. There are nowadays several proposals relying on different approaches often aiming at giving a score instead of a yes/no. One of the most well-known meters is "zxcvbn" [26] in use in the Dropbox service and probably in many others as it is an open-source solution. Some meters rely on cracking techniques to assess the probability the password would be produced by such techniques, such as the OMEN-based solution [2] or the PCFG-based one [8,20]. Some meters are machine-learning based such as the neural network based meter of Melicher et al. [14] extended by Ur et al. [17], or the multi-modal approach of Galbally et al. [5]. While increasing the security of digital services by enforcing users to select strong and safe passwords, those meters also play a role in the analysis of password datasets to better understand human tendencies but also classifying passwords in classes of strength.

3 How Can Quality Be Measured?

The definition of a metric to measure and classify the quality of wordlist given as input to password cracking process is a difficult one. The expected features a wordlist should have, and be evaluated on, is likely to vary depending on the final cracking process and its context. The particular scenario of the attack, such as whether it is a targeted attack to a specific individual or a fishing attack that targets a group of people plays a role in the approach we take for creating a wordlist. Other factors, such as the language of the target(s), the type of service, etc., also have to be taken into account, since the approach will be different [22].

Therefore, creating a single metric for measuring the quality of dictionaries is not suggested. Instead, we are looking at a number of factors that can be taken into account, alone or combined, when deciding the type and makeup of a wordlist that is likely to make it the optimal candidate for a specific scenario. These factors are presented below.

3.1 Final Percentage of Passwords Cracked

This is the most straight-forward metric in password cracking, where a wordlist is evaluated based on the amount of passwords it has cracked from a target list. This metric is typically the most important one especially in the case where the concern is the volume of cracked passwords and the focus is not on a single target or a small number of targets.

Some password cracking processes have a fixed limit of candidates they can generate based on the size of the input wordlist. For example, a straight dictionary attack will generate as many candidates as there are in the wordlist, potentially multiplied by the number of mangling rules, if they are used. Some other processes can be considered as endless, such as for example Markov-based ones if they are not limited, and will continuously produce candidates like an unbounded exhaustive search would do. As a consequence, those endless processes would theoretically always retrieve 100% of the passwords if they are given enough time which in most cases is not practical. That is why it is necessary to set a limit to the number of candidates that a process is allowed to generate and test. Such limit can be adjusted depending on the complexity of the scenario we want to assess.

3.2 Number of Guesses Until Target

The previous metric alone is not enough to evaluate a wordlist as other factors can be relevant in some scenarios. For example, one wordlist may recover 75% of passwords while a second may get only 60%. But, it might be the case that the second one reached a score of 50% with less candidates generated than the first one. In some scenarios, the number of candidates that can be evaluated in a reasonable time is strongly limited because of hardware constraints or high complexity of the underlying function making the second wordlist more interesting for a particular scenario. Assessing the number of guesses needed to reach a targeted percentage of retrieved passwords can help to select a wordlist more suited to the conditions of some scenarios.

3.3 Progress over Time

Another metric that is strongly related to the amount of found passwords, is the pace in which the passwords are retrieved. During password cracking, an updated percentage of results at pre-established checkpoints, can give us insights into the performance of the dictionary over time. For example, at some point in

the cracking progress, the amount of new passwords guessed at every checkpoint might start decreasing, which means that the new password candidates that are checked do not recover new passwords anymore. This is often another criterion to stop the process and also a hint, for dictionary lists that are ordered by count, that the size of the input wordlist can be decreased without a remarkable effect on performance. This criterion is the second derivative of the curve of found passwords over number of guesses and represents a process with an upper bound, compared to the metric outlined in Sect. 3.1.

3.4 Size of Wordlist

Closely related to the stop criterion of incremental progress over time, the size of the wordlist is another metric that can be taken into account. For example, when two wordlists, with a significant difference in size, produce similar numbers of cracked passwords, the smaller wordlist can be thought of as of better quality, as it needs less information to achieve the same results. From another point of view, when the foreseen process is machine learning based, a larger wordlist could be preferred to reinforce the training phase.

3.5 Better Performance with Stronger Passwords

Another metric that should be considered is the performance of a wordlist against difficult passwords. For example, if two wordlists are similar in the previous criteria, i.e. crack about the same number of passwords, do it at about the same amount of time and are of similar size, the one that cracks more difficult to recover passwords is stronger, and should be assigned a higher score. Often, in real world scenarios and if the hash function permits it, an exhaustive search is performed first for the weaker passwords. This means that a wordlist that performs well against passwords that cannot be recovered by a brute force attack, is more valuable.

3.6 Compound Metric

The above metrics, cannot accurately assess any individual wordlist. Focus on one, or more of the above is necessary, according to the target case. For example, when we are concerned with recovering as many passwords as possible, the percentage of success is what matters most. But, when we want the largest number of passwords in a specified amount of time, the trade off between success and time is important. When we focus on a single target, or a small number of targets, like during the course of an investigation, speed and possibly the performance against stronger passwords are important factors to consider. This criterion can be refined to look at the number of guesses needed to retrieve a given percentage of passwords of a certain strength class as defined later in Sect. 5.

Furthermore, it has been shown that large corpora of passwords obey Zipf's Law, meaning that the frequency of each popular password, would be inversely

proportional to each rank, i.e. the second most popular password would appear approximately half as many times as the first [21]. According to this analysis, the level of fit of a particular dataset under this model, could be an indicator of strength of the dataset.

This brings forward the need for a compound metric, one that combines two or more of the above criteria, to get an evaluation tailored to a specific case.

4 Methodology

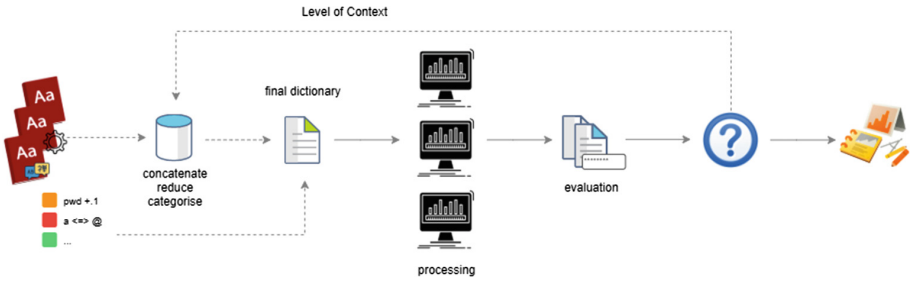

Fig. 1. Methodology

Based on the metrics analysed in Sect. 3, the evaluation of input wordlists, with the purpose of arriving at the optimal one, is a case by case scenario and is based on the individual needs, be it rate of success, time it takes to achieve a certain threshold or success at recovering one specific strong password.

Furthermore, the success of the input wordlist is not only based on the above factors, but also on the tools we use to do the password cracking. For example, PCFG works better and estimates more accurate probabilities when the input dictionary does not contain only unique entries, but repeated ones. However, such dataset with repetitions are rarely available to the research community. Therefore, in order to test wordlists, it is essential to have a few different tools to evaluate them with.

For this purpose, we have chosen to use four tools; John the Ripper, Omen, PCFG and Prince. The aim of this is not to compare these tools and find which is the better one, rather to make sure we have compared input dictionaries as thoroughly as possible. In order to perform this part of the process, we are using the Password Guessing Framework (PGF). This tool is an open source tool to automate the process of comparing different password guessers. As already mentioned, the reason we are using this tool is not to compare password guessers, rather to avail of its ability to automatise the setting up of the cracking process. Indeed, PGF allows the setting up of 'jobs' which will be processed sequentially, where you can define the parameters of the guessing tool, input dictionary, target

list (hashed or plain), max number of guesses, etc. The results come in the form of `*.txt` and `*.csv` files containing an analysis of the number of found passwords, a list of those as well as data on cracking performance over time. All this information is then used by us for the evaluation of the input dictionaries.

As the methodology flowchart on Fig. 1 shows, the performance of the different input dictionaries is evaluated and presented or fed back to pre-processing. The pre-processing step, which is outside the scope of this paper and part of our future work, contains the creation of tailored input lists from existing or custom dictionaries, the tailoring of mangling rules to the specific scenario (keeping in mind whether the end goal is success ratio, time efficiency, recovery of a targeted password, etc.). The feedback from the evaluation process will re-trigger the wordlist creation process in order to modify the size of the list, the number and quality of mangling rules and the level of contextual information, with the end goal being to optimise the generation of a password candidate list.

The goal of this framework will be the evaluation of all created password candidate lists under the same scenarios and by taking into account the metrics discussed in Sect. 3, to arrive at the optimal wordlist for each scenario. In the next section, we present a preliminary analysis into how the evaluation process of the framework works, what results we can get and their significance.

In this preliminary experiment, the main focus is assessing password candidates that stem from leaked databases, to see whether a wordlist that is thematically similar to the list of passwords we want to crack can yield better results than a generic wordlist.

5 Preliminary Analysis

5.1 Dataset Selection and Creation

The datasets we used to conduct a preliminary experiment to gauge the role of context in password selection, can be summed up in Table 1. The dataset we have named Comb4 is a combination of four different leaked datasets from four categories, music, cars, videogames and manga. Our aim was to create a combination of datasets from different sources to cover a wide spectrum of user interests and ascertain whether the purpose of the forum for which the password is created for, plays a role during the creation process. Because of availability, the evaluation datasets are from only two of the above categories, manga and videogames. We have also used RockYou as a baseline to compare the other datasets with. The version of RockYou we used is the one with 14 million unique passwords but the full version of 32 million passwords was also tested with PCFG, but yielded to slightly better but similar results. All the datasets used in this experiment stem from an online database of leaked wordlists named *hashes.org*, the use of which has been reviewed and approved by the Office of Research Ethics of University College Dublin.

Table 1. Datasets in use

	Dataset	Size
Comb4	axemusic	252,752
	jeepforum	239.347
	minecraft	143,248
	mangatraders	618,237
Evaluation	boostbot	143,578
	mangafox	437,531
	RockYou	14,344,391

5.2 Example Use Case

In this example use case, the evaluation datasets, Boostbot and Mangafox, as well as RockYou are used without modification with all four password crackers and 10 billion candidates were generated and evaluated for each process. The results of the cracking progress over time for RockYou, Mangafox and Boostbot can be found in Figs. 2, 3 and 4 respectively. As can be seen on all three figures, PCFG performs better for all three datasets and especially in the case of RockYou the result is much more distinguished. Comb4 contains `1,253,531` passwords, of which `1,096,481` are unique. Of these, RockYou PCFG managed to crack more than 60% (`768,341`) or in case of unique passwords `617,016`. This result is significantly more than the other three guessers, but also significantly more than Mangafox and Boostbot. In fact, RockYou performed better than both of those datasets with all four guessers.

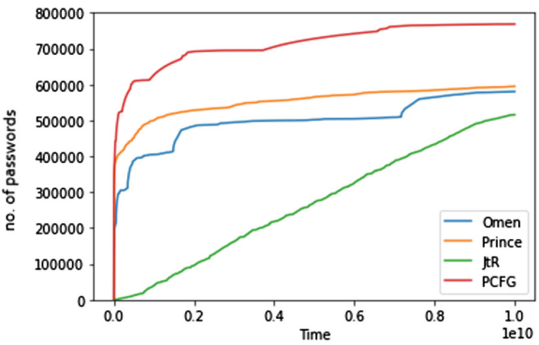

Fig. 2. Cracking progress over time with `RockYou`

This result is not a surprise because the first key difference between RockYou and Mangafox and Boostbot is their size, as seen in Table 1. RockYou is about 32 times larger than Mangafox and 100 times larger than Boostbot. This means

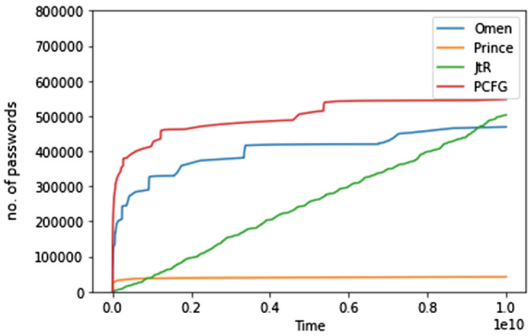

Fig. 3. Cracking progress over time with `Mangafox`

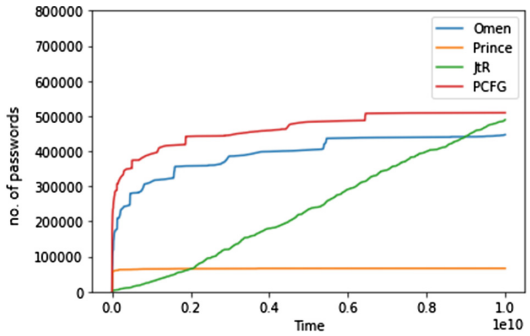

Fig. 4. Cracking progress over time with `Boostbot`

that there is certainly more diversity in the password candidates generated with rockyou. If then we focus only on Mangafox and Boostbot, Mangafox performed slightly better, which can be on account of its larger size but also on the fact that the largest dataset in Comb4 is Mangatraders, which is also another manga related leak.

Furthermore, we can see that Prince under performed with the smaller datasets, while it had the second best performance with RockYou. This is due to the principle of PRINCE combining entries of the input dictionary to create new candidates. The input in the two smaller dataset are more sophisticated than those in Rockyou. There, their concatenation leads to very complex candidates with a low probability of being in the targeted list. A pre-processing could be apply in PRINCE to better integrate such type of input wordlist.

JtR on the other hand, steadily improved throughout the cracking process, almost reaching PCFG towards the end for both Mangafox and Boostbot.

Table 2. Strength distribution in Comb4

	Comb 4	Axemusic	Jeepforum	Mangatraders	Minecraft
Class 0	46,645	4,143	34,832	6,471	1,199
Class 1	503,809	93,100	128,279	241,745	40,685
Class 2	395,202	87,158	58,189	205,218	44,637
Class 3	226,243	55,395	16,657	118,840	35,351
Class 4	81,624	12,898	1,388	45,962	21,376

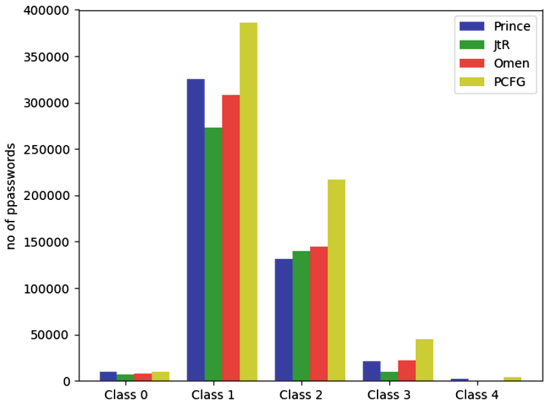

Fig. 5. Strength of cracked passwords with `RockYou`

Because the amount of cracked passwords, as mentioned in Sect. 3 cannot be the only metric to take into account - otherwise RockYou would have been the clear winner - we also looked at the strength classes of the cracked passwords by each dataset.

In order to evaluate that, we used `zxcvbn`, as referenced in Sect. 2. With `zxcvbn` passwords are divided into 5 classes, according to their strength, i.e. how well they would withstand a cracking attack, with class 0 being the least secure and class 4 being the most secure. This classification takes into account rules set by common password policies but also l33t speak, common passwords and patterns to make a determination. Table 2 shows the classification of Comb4 in these classes and Figs. 5, 6 and 7 show the cracked passwords per class for RockYou, Mangafox and Boostbot respectively, with all four password guessers.

As can be seen by the above figures for all three datasets, the distribution of found passwords follows the distribution amongst classes of the Comb4 dataset which can be seen in Table 2. For RockYou, we can see that PCFG as expected performed better on all classes (except class 0, where all four are on par) with an especially big difference for class 3 and 4 compared to the other guessers. When it comes to Mangafox, other than Prince, the performance was similar for the three other guessers. Interestingly, Mangafox and Boostbot were able to find

about one third as many passwords in class 4 as RockYou with PCFG, especially considering the big difference in size. Even more remarkably, MangaFox and Boostbot outperformed RockYou in case of class 4 with both JtR and Omen.

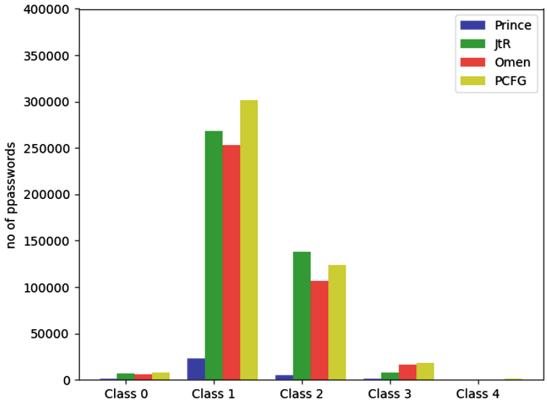

Fig. 6. Strength of cracked passwords with `mangafox`

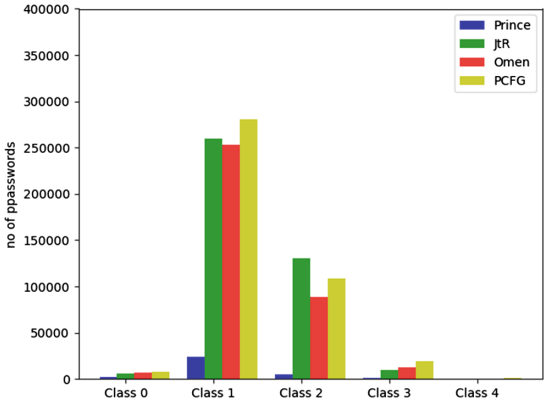

Fig. 7. Strength of cracked passwords with `boostbot`

Oftentimes, in real world scenarios, the way to go would not be to choose one password cracking guesser or input wordlist over the other but stack them. For this reason we wanted to see how using a big input dataset like RockYou could be complemented, rather than beat. Table 3 shows the number of unique passwords that were found only by Mangafox and Boostbot and not by RockYou, in total and also their distribution amongst the 5 classes of `zxcvbn`.

Table 3. Passwords found by Boostbot and Mangafox but not by RockYou

		JtR	Omen	Prince	PCFG
All	Boostbot	26,109	32,911	5,788	17,811
	Mangafox	26,694	37,977	3,608	22,121
Class 0	Boostbot	182	265	22	73
	Mangafox	210	227	35	109
Class 1	Boostbot	11,960	16,698	3,095	3,659
	Mangafox	12,005	14,603	1,664	5,393
Class 2	Boostbot	10,439	11,628	1,730	8,303
	Mangafox	12,192	16,702	1,476	11,303
Class 3	Boostbot	3,512	4,171	796	5,266
	Mangafox	2,285	6,325	373	4,868
Class 4	Boostbot	16	149	145	510
	Mangafox	2	120	60	448

As can be seen in Table 3, this is a substantial addition of found passwords. In fact, with PCFG, the addition of either the passwords recovered by Mangafox or Boostbot, brings a 14% increase to the total, which is an important addition, being that this is the class of passwords that is the least easy to recover. Even in the case of Prince that generally underperformed, 73% for Mangafox and 63% for Boostbot, of the passwords that were found with these two datasets, were not recovered by RockYou. The recovery of class 4 passwords with Omen was even more impressive because about twice as many passwords were recovered with Mangafox or Boostbot compared to RockYou. Finally, even in the case of JtR, with a meagre 2 passwords recovered from Mangafox and 16 from Boostbot, RockYou did not manage to find any of class 4.

5.3 Breakdown of Comb4

In order to assess the quality of the input wordlists even further, we decided to break down Comb4 into the four individual datasets it was generated from, axemusic, jeepforum, mangatraders and minecraft. The breakdown of these datasets to zxcvbn classes is shown in Table 2. Additionally, Table 4 shows the amount of passwords found of each dataset of Comb4, by each input wordlist for all four password guessers. The result that pops up is that in the case of Minecraft, and excluding the underperforming Prince, the amount of passwords found by RockYou, Mangafox and Boostbot are very similar. A possible explanation of these results is that the thematic proximity compensates for the difference in size. In fact, Boostbot is the smallest dataset (about 100 smaller than RockYou) but thematically is the one closest to Minecraft. And Mangatraders is still a lot more relevant to Minecraft than for example Jeepforum.

Still, we see that for the other three datasets, RockYou performs a lot better than Mangafox and Boostbot. Even in the case of Mangatraders, while the

Table 4. Breakdown of Comb4

		JtR	Omen	Prince	PCFG
Axemusic	RockYou	60,583	86,417	88,776	131,485
	Mangafox	57,923	67,934	6,456	93,843
	Boostbot	57,120	63,969	8,027	86,090
Jeepforum	RockYou	96,894	105,232	109,847	133,665
	Mangafox	93,250	74,535	7,461	92,265
	Boostbot	89,084	72,753	6,966	83,477
Mangatraders	RockYou	267,553	289,903	299,890	373,483
	Mangafox	260,126	234,834	18,067	255,964
	Boostbot	252,338	221,328	22,121	241,774
Minecraft	RockYou	39,050	42,630	36,335	55,226
	Mangafox	41,417	43,171	3,561	50,221
	Boostbot	40,624	39,345	5,741	43,953

results for JtR and Omen are close, RockYou's performance for PCFG is significantly better, since for PCFG the full RockYou list of 32 millions was used, so that PCFG can take advantage of repetitions of passwords to form better probabilities.

In this case, the size of the input wordlist makes the difference and this along with percentage of success is the one to watch. Still, as mentioned above, in real cases the goal is not to choose one wordlist over the other but to complement it. For this reason, we are looking again at another metric, performance for stronger passwords. In Table 5 we are looking at PCFG only, and focusing on the class 3 and class 4 passwords that were recovered by only Mangafox and Boostbot and not by RockYou. As can be seen, the percentage of passwords found by these two datasets and not RockYou was significantly higher for minecraft and mangatraders, the two datasets that were contextually closer to mangafox and boostbot.

Table 5. Breakdown by dataset for PCFG, Class 3 and Class 4

PCFG	Mangafox		Boostbot	
	Class 3	Class 4	Class 3	Class 4
axemusic	1.3%	0.4%	1.6%	0.5%
jeepforum	0.9%	0.6%	0.9%	0.4%
mangatraders	2.2%	0.7%	2.7%	0.9%
minecraft	4.1%	0.3%	3.4%	0.2%

6 Discussion

The question that arises from the preliminary results is, what do these two datasets have that RockYou does not? Both Mangafox and Boostbot stem from online leaks and there is no processing, augmentation or other customisation done to them. Furthermore, their size is small compared to the 14 million of RockYou passwords (32 million in the case of PCFG). The one advantage these datasets have, is that thematically they are closer to the target datasets.

Overall, the performances between the three wordlist are comparable when considering the JtR approach and close when considering with Omen, both of which are Markov based models. The results are really poor when it comes to Prince (for Mangafox and Boostbot) but a pre-processing on the wordlist to make a better usage of it could modify those results. PCFG works better than the other processes but with a clear advantage for RockYou. This is probably thanks to the difference of size, giving more chance to PCFG to infer and reuse the grammar. One way to possibly improve the results of PCFG with Mangafox and Boostbot could be to reuse the grammar trained from RockYou but feeding the special list with the content of the other dataset.

Both Mangafox and Boostbot have a better ratio of passwords found in Class 3 for Minecraft and in a less impressive manner on Class 4 for Mangatraders. The found passwords are significantly fewer for Axemusic and Jeepforum, probably due to a lesser proximity of the communities. Surprisingly, Mangafox performs better than Bootsbot on Minecraft and Boostbot better on Mangatraders than Mangafox, while we would have expect the results to be the other way around. Still, the communities of manga and videogames are more closely associated with each other than music and cars, so this close proximity might be the explanation. Finally, while Mangafox performs poorly on Class 3 of Jeepforum, it performs relatively well, even if the numbers are small, on Class 4.

7 Conclusion and Future Work

The developed framework provides a new methodology to assess and compare wordlists. It highlights that wordlists behave differently depending of the context of the target dataset and it can therefore be used to develop and assess wordlist generation processes in several scenarios. Focusing on the different classes of strength is also useful to evaluate the quality of wordlists to retrieve stronger passwords.

Our analysis highlighted also that the size and the composition of the wordlists have a strong impact on some processes, for example Prince and PCFG, while it is less visible for some other processes. Dedicated pre-processing shall be envisaged to better prepare the wordlists for those processes.

Therefore, it is clear that not one metric can stand alone, evaluate a wordlist thoroughly and assign a score that can predict how well that wordlist will do against a target. As was the case in this preliminary analysis, a compound metric will be needed for the evaluation, and even then, there should be room left for its parameterisation for each attack scenario.

Our future work will be twofold. On one hand it will focus on the design of a process to generate targeted wordlists which will be tailored to specific scenarios, with the aim to achieve better results for some targeted community or classes of strength. This process will involve the customisation of the wordlist based on contextual information known about the target, as well as by a constant feedback process with which the levels of contextualisation, the choice of mangling rules and the size of the input wordlist, will be optimised. Secondly, through the process mentioned above, further work will be conducted to assess and combine the metrics presented in this paper, and optimise them for the evaluation of wordlists.

References

1. Burr, W.E., Dodson, D.F., Polk, W.T.: NIST special publication 800–63 - electronic authentication guideline. Technical report, National Institute for Standards and Technology (2004)
2. Castelluccia, C., Dürmuth, M., Perito, D.: Adaptive password-strength meters from Markov models. In: NDSS (2012)
3. Du, X., et al.: SoK: exploring the state of the art and the future potential of artificial intelligence in digital forensic investigation. In: Proceedings of the 15th International Conference on Availability, Reliability and Security. Association of Computing Machinery (2020)
4. Dürmuth, M., Angelstorf, F., Castelluccia, C., Perito, D., Chaabane, A.: OMEN: faster password guessing using an ordered Markov enumerator. In: Piessens, F., Caballero, J., Bielova, N. (eds.) ESSoS 2015. LNCS, vol. 8978, pp. 119–132. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15618-7_10
5. Galbally, J., Coisel, I., Sanchez, I.: A new multimodal approach for password strength estimation-part I: theory and algorithms. IEEE Trans. Inf. Forensics Secur. **12**(12), 2829–2844 (2017)
6. Haque, T., Wright, M., Scielzo, S.: Hierarchy of users' web passwords: perceptions, practices and susceptibilities. Int. J. Hum Comput Stud. **72**(12), 860–874 (2014)
7. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: PassGAN: a deep learning approach for password guessing. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 217–237. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_11
8. Houshmand, S., Aggarwal, S.: Building better passwords using probabilistic techniques. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC 2012, pp. 109–118. Association for Computing Machinery, New York (2012)
9. Kanta, A., Coisel, I., Scanlon, M.: A survey exploring open source intelligence for smarter password cracking. Forensic Sci. Int. Digit. Investig. **35**, 301075 (2020)
10. Kanta, A., Coisel, I., Scanlon, M.: Smarter password guessing techniques leveraging contextual information and OSINT. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–2. IEEE (2020)
11. Kanta, A., Coray, S., Coisel, I., Scanlon, M.: How Viable is Password Cracking in Digital Forensic Investigation? Analyzing the Guessability of over 3.9 Billion Real-World Accounts. Forensic Science International: Digital Investigation, July 2021

12. Kuo, C., Romanosky, S., Cranor, L.F.: Human selection of mnemonic phrase-based passwords. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, pp. 67–78. Association for Computing Machinery, New York (2006)
13. Liu, Z., Hong, Y., Pi, D.: A large-scale study of web password habits of Chinese network users. JSW **9**(2), 293–297 (2014)
14. Melicher, W., et al.: Fast, lean, and accurate: modeling password guessability using neural networks. In: Proceedings of the 25th USENIX Conference on Security Symposium, SEC 2016, pp. 175–191. USENIX Association, USA (2016)
15. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_36
16. Shay, R., et al.: Designing password policies for strength and usability. ACM Trans. Inf. Syst. Secur. **18**(4), 1–34 (2016)
17. Ur, B., et al.: Design and evaluation of a data-driven password meter. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI 2017, pp. 3775–3786. Association for Computing Machinery, New York (2017)
18. Ur, B., et al.: "I added '!' at the end to make it secure": observing password creation in the lab. In: Eleventh Symposium on Usable Privacy and Security (SOUPS 2015), pp. 123–140 (2015)
19. von Zezschwitz, E., De Luca, A., Hussmann, H.: Survival of the shortest: a retrospective analysis of influencing factors on password composition. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013. LNCS, vol. 8119, pp. 460–467. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40477-1_28
20. Wang, D., He, D., Cheng, H., Wang, P.: FuzzyPSM: a new password strength meter using fuzzy probabilistic context-free grammars. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 595–606 (2016)
21. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. **12**(11), 2776–2791 (2017)
22. Wang, D., Wang, P., He, D., Tian, Y.: Birthday, name and bifacial-security: understanding passwords of Chinese web users. In: 28th USENIX Security Symposium (USENIX Security 2019), pp. 1537–1555. USENIX Association, Santa Clara, August 2019
23. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: an underestimated threat. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 1242–1254. Association for Computing Machinery, New York (2016)
24. Wash, R., Rader, E., Berman, R., Wellmer, Z.: Understanding password choices: how frequently entered passwords are re-used across websites. In: Twelfth Symposium on Usable Privacy and Security (SOUPS 2016), pp. 175–188. USENIX Association, Denver, June 2016
25. Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 391–405. IEEE (2009)
26. Wheeler, D.L.: zxcvbn: low-budget password strength estimation. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 157–173 (2016)