Crack Me I'm Famous: cracking weak passphrases using publicly-available sources

Hugo Labrande
hugo.labrande@inria.fr

INRIA Nancy, Equipe CARAMEL

University of Calgary ISPIA

Abstract. Using only publicly-available data and very modest computing power, we built a dictionary of several million famous sentences, and used it to crack millions of passphrases, and some mnemonic-based passwords. It shows that even low-skilled attackers can crack such passwords very easily, confirming that they offer very poor security.

1 Introduction

Password-based authentication requires users to pick passwords that are complex to guess but easy to remember; in practice, a majority of them do not fulfill both criteria. For instance, some users still choose to pick one common word (e.g. "password"), which is easily found with a dictionary attack, or the name of a brand, or a slang word, which are not that much harder to find with bigger dictionaries [11,6]. Another strategy is to pick a simple password and complexify it by mangling it; however the methods chosen by users are often very simple (e.g. add a digit at the end, replace "o" by "0", etc.), and hackers can crack them with lists of common word-mangling rules, that are all applied to every word in the dictionary by password cracking software.

Another strategy that sometimes implemented is to ask users to choose a passphrase, i.e. a random sequence of words; one can also use mnemonic-based passwords [9], which consists in making short passwords out of passphrases. The rationale is that passphrases are longer than passwords, but easier to remember. Passphrase-cracking has been less studied than password-cracking, maybe because passphrases are not as widely used. Several works [9,5] have attempted to build sentences from a sequence of words by analyzing how users create a random sentence. One can also use sentences from a corpus of "real text" as potential passphrases, like [12], who used every book from the Gutenberg project and every Wikipedia article. This yielded about 1.3 billion passphrases, allowing to crack passwords like "thereisnofatebutwhatwemake" or "givemelibertyorgivemedeath".

Our work attempts to study the use of famous sentences, which is to say sentences appearing in mainstream culture (e.g. quotes, slogans, proverbs, expressions, etc.) as opposed to user-crafted and user-specific sentences, as passphrases. We do so because some studies [9,5] showed a majority of users tend to choose famous sentences for their passphrases or mnemonic-based passwords; some passwords found by [12] seem to confirm this (e.g. the two examples given above). We isolated 65 million famous sentences from publicly-available corpora in order to build a dictionary which allowed us to crack more than 4 million passwords. Most of them can probably be cracked using the methods of [12]; however, we emphasize that our method is simpler, faster and requires less data and memory, which means that it is within the grasp of any attacker; as such, our work confirms the well-known fact that such sentences offer poor password security. Also, system administrators can use our methods and our dictionaries to find vulnerable, but not too obvious, passwords in their network.

2 Building dictionaries

In what follows, we focus only on passphrases, and disregard all individual words, since there are already many wordlists containing millions of words, e.g. [11,7]. Hence, we hope our dictionaries will be orthogonal to existing wordlists, with few overlaps in the passwords guessed.

We first used the list of titles of Wikipedia articles, which is conveniently directly downloadable [4]. We considered the English, French, Spanish, German, and Italian versions to cover a great number of sentences and expressions; we then merged the five resulting dictionaries. Our process was as follows: only keep entries containing a space, remove disambiguation terms (e.g. "Queen (band)"), then for each entry with a leading article [1], add the same entry with no article. We preserved original punctuation since word-mangling rules allow us to try a password with or without punctuation. In the end, we got a first dictionary, called **wikititles**, with 12,925,291 entries, containing more than 1 million non-English entries.

We also used Wikiquote to add quotes that do not have their own articles on Wikipedia. We first downloaded the complete contents of five Wikiquote projects (English, French, Spanish, Italian, German) [4], then used a few scripts to pick out the quotes [2] and remove XML and Wikipedia tags: those scripts are not very hard to write and one can extract most

1. We used "The", "Le", "La", "Les", "El", "Los", "Las", "Der", "Die", "Den", "Das", "L'", "Il", "Un", "Una", "Une", "A", "An", "Ein", but missed a few like "Des" or "Gli".
2. For instance, most of them are on lines starting with an asterisk.

quotes with a bit of care. We also added each quote's individual sentences to our dictionary. The result was what we call **wikiquote** dictionary, made of 2,159,691 entries.

Those dictionaries still miss good candidates for passphrases; for instance, a song that was not released as a single may not have its own Wikipedia page, but should still be considered (e.g. "afraidtoshootstrangers" [12]). We noticed that quotable group of words often appear either between quotation marks, or in bold (two single quotes in Wikipedia syntax) or italics (three single quotes); hence, we created the **wikiarticles** dictionary by capturing groups of words which appear between such delimiters in the body of any Wikipedia article. Once again, this only required a few simple scripts, which ran on the five Wikipedias we consider here in no more than a dozen hours; this yielded 56,910,550 entries, which is about the same as other cracking dictionaries (e.g. [7,11]). We note that 60% (10 million candidates) of the **wikititles** and **wikiquote** dictionaries does not appear in this dictionary, which means those are still useful.

We also considered other sources of data in the public domain to build smaller, more targeted, dictionaries. We used the Rap Dictionary [1], a wiki dedicated to hip-hop culture; most of its pages are spam, but we reasoned that spam may be used for unique passwords by some users (e.g. "cheapviagra"). We manually grabbed the page titles, which gave us the dictionary **rapdict** using 2,200 pages specifically about hip-hop, and **rapdict+spam** using all 20,000 pages, including spam. We also used the Free Online Dictionary Of Computing (FOLDOC [1]), a database of 15,000 computer science terms and slang; keeping only the terms which contained a space gave the dictionary **foldoc**, which has 6,600 entries. Most terms present in those dictionaries do not appear in our other dictionaries.

Many more sources of data could be exploited, even if scraping them may violate Terms of Service. We avoided them to show that someone (e.g. a sysadmin) could build a good dictionary using only legal sources; furthermore, only Wikipedia offers direct download. Examples of interesting sources are the Urban Dictionary, Twitter (whole tweets, hashtags), message boards (signatures, usernames), IMDB, song lyrics websites, etc. A few of them appear to have been used by [12] or [6], although we do not know how useful they were.

3 Results

As a target for our attack, we picked the MD5 hashes from the Korelogic dump [8], which contains 139 millions hashes, around 85% of which have been cracked [10]. We used John the Ripper Jumbo [2]; unlike ocl-Hashcat [3], it only uses CPU power instead of GPU, which leads to modest but very reasonable speeds. Furthermore, we used word-mangling rules to increase the number of passwords we cracked, using the **Wordlist** set of rules bundled with JtR, and 91 passphrase-mangling rules that we created, which all include a transformation on whitespace (unlike **Wordlist** rules); see the discussion in Section 5. The computation took a few weeks on an outdated laptop with a 2GHz processor: dedicated crackers with several high-speed GPUs could probably replicate our results in a few hours.

Name of dictionary	Entries	Pwd cracked
wikititles	12,925,291	1,242,879
wikiquote	2,159,691	435,744
wikiarticles	56,910,550	3,924,023
rapdict+spam	20,017	7,778
rapdict	2,213	3,146
foldoc	6,589	6,529
merged	65,351,745	4,266,871

Table 1. Results

It appears that our dictionaries allow us to crack more than 4.2 million passwords derived from famous sentences, which is number; around 1.8 million of them have 10 characters or more, and around 200,000 more than 16 characters. Furthermore, our dictionary can supplement existing wordlists: for instance we found that 1.93 million passwords, or 40% of our total, cannot be cracked using the **GDictv2** [7] wordlist, a popular wordlist made of over 21 million candidates. Our other sources gave interesting results, since several hundred unique passwords were obtained with those dictionaries (except **rapdict**, which did not crack any new passwords).

wikititles	**Lamborghini Murciélago LP 670-4 SuperVeloce** **Ghost in the Shell: S.A.C. Solid State Society**
wikiquote	**Be sure to drink your ovaltine**, **judge a man by his questions rather than by his answers** *(tied for longest password ever cracked with "LyingIsTheMostFunAGirlCanHaveWithoutTakingHerClothesOff" by [6])*
wikiarticles	**superlativeextruderinterlockedtechnologyexclusivenexus** **Wikipedia, l'encyclopedie libre et gratuite** **I could tell you but then I'd have to kill you** **Pa's wijze lynx bezag vroom het fikse aquaduct** **Cantami o diva del pelide Achille l'ira funesta**
foldoc	**exercise, left as an** **polymorphic lambda-calculus**
rapdict+spam	**fastcashadvanceonline** **penisenlargementthatworks**

Table 2. Examples of cracked passwords

4 Mnemonic-based passwords

The study [9] showed that a majority (50%-65%) of users choose a famous sentence when asked to construct a mnemonic-based password. We built a dictionary of 33 million mnemonic passwords based on famous sentences, by taking the first letter of each word of a phrase, which is a common method [9]; one could also look at leet-speak or homophonic substitution (e.g. "@" for "at") [9] but we did not. We kept punctuation and capitalization, and used the same rules as with the other dictionaries.

Analyzing the results for this part was rather hard: trivial passwords (e.g. "qwertyuiop", "111111") were produced, as well as single words which were in fact the initials of a passphrase; hence we would recommend to only use this dictionary on hashes remaining after using a standard wordlist, which we did not attempt for lack of time. Nonetheless, the number of mnemonic-based passwords we were able to isolate was frustatingly small, which could mean that the technique is not used very widely, or that more sophisticated rules are used. However, by only focusing on the longest passwords, we found more than 300 passwords which looked hard to guess by conventional techniques; guessing the underlying passphrases was done manually, using the fact that order was preserved when creating our mnemonic-based dictionary. Hence, we report cracking a 19-character mnemonic-based password, "1lomtjjzictcttsdkcs" ("Pan Tadeusz"), as well as "uuddlrlrababsS" ("Konami cheat code"), "tbtitbtwtbtewb" (catchphrase of Bret "The Hitman" Hart), and a few dozen others.

5 Passphrase-mangling rules

We attempted to determine which word-mangling rules work best in the case of dictionaries of passphrases, to attempt to optimize password crackers: for instance only one rule in the set of standard rules present in John the Ripper takes whitespace into account. We generated by hand a set of rules targeting passphrases, using the following behaviors: "delete spaces" or "replace spaces by underscores", and any combination of "delete punctuation", "delete punctuation except ?,! and .", "delete symbols", "lowercase", "capitalize", "uppercase", and "append digits". This created 91 new rules, which we combined to **Wordlist** rules to get the results of Section 3 and 4. We then tested the individual efficiency of each rule.

Our results show that "delete spaces, lowercase" and "delete spaces, append digit" are by far the best rules, followed by "delete spaces and punctuation, append digit", "delete spaces, append digit, lowercase or uppercase" and "delete spaces". We note the underwhelming performance of all rules replacing whitespace by underscores. We recommend the rule "delete spaces, append digit" to be added to **Wordlist** rules, since it is as efficient as "delete spaces, lowercase", which is already present. Finally, it would be interesting for passphrase-cracking to be able to capitalize the first letter of each word, which JtR does not seem to allow.

6 Conclusion

We showed new techniques to build passphrase-cracking dictionaries using freely available sources of data and a few scripts; our method is fast and efficient, and yielded several million candidates. We cracked a great number of hashes using our dictionaries, yielding long and complicated passwords, e.g. a 55-character long password and a 19-character long mnemonic-based password. Finally, we also studied which passphrase-mangling rules were the most useful, and gave a few recommendations to password crackers. Our work shows any low-skill, low-resource attacker can crack interesting passphrases that probably look secure to users; our work could be refined by studying more sources of data, or including more word-mangling rules to the set of rules considered.

References

1. Foldoc & Rap Dictionary. `http://foldoc.org/` & `http://www.rapdict.org/`.
2. John the Ripper. `www.openwall.com/j`.
3. oclhashcat. `http://hashcat.net/oclhashcat/`.
4. Wikimedia dumps. `http://dumps.wikimedia.org`.
5. J Bonneau and E Shutova. Linguistic properties of multi-word passphrases. In Springer Berlin Heidelberg, editor, *Financial Cryptography and Data Security*, pages 1–12, 2012.
6. Y. Chrysanthou. I have the ♯cat so I make the rules. Passwords14, 2014.
7. Gdataonline. `http://gdataonline.com/downloads/GDict/`.
8. Korelogic. `https://www.korelogic.com/InfoSecSouthwest2012_Ripe_Hashes.html`.
9. C. Kuo, S. Romanosky, and L.F. Cranor. Human selection of mnemonic phrase-based passwords. In ACM, editor, *Second Symposium on Usable Privacy and Security*, pages 67–78, 2006.
10. m3g9tr0n. Cracking story : How i cracked over 122 million sha1 and md5 hashed passwords. Thireus's blog.
11. Sébastien Raveau. Hunting for passwords or putting an end to the arms race. Passwords12, 2012.
12. K. Young and J. Dustin. Password cracking, from "abc123" to "thereisnofatebutwhatwemake". Passwords13, July 2013.