The Benefits of Understanding Passwords

Markus Jakobsson
PayPal

Mayank Dhiman
PEC University of Technology

Abstract

We study passwords from the perspective of how they are generated, with the goal of better understanding how to distinguish good passwords from bad ones. Based on reviews of large quantities of passwords, we argue that users produce passwords using a small set of rules and types of components, both of which we describe herein. We build a parser of passwords, and show how this can be used to gain a better understanding of passwords, as well as to block weak passwords.

1 Introduction

A good password is hard to guess. Conversely, of course, a bad password is *easy* to guess. But what is it that makes something hard to guess, and how can we tell? Traditional wisdom holds that passwords should be longhand complex. Password strength checkers typically translate this into minimum length requirements and demands to include an uppers case, lower case character, and a numeral. As a result, a large number of users – somewhat predictably – starts their password with an upper case, continue with lower case throughout, and append the digit 1 at the end. This predictability in terms of user behavior makes passwords easier to guess.

It is easier to tell that something is *easy* to guess than that it is hard to guess. For example, the following potential passwords are easy to guess: *fraternity* (a dictionary word); *$L* (a very short string); *qwertyuiop* (a string with a very predictable pattern); and *LoveLoveMeDo* (famous lyrics). Similarly, one can look at the commonality of passwords [15] – any user who wants to use a password that has already reached the limit has to think of another password. We can make a long list of reasons to consider a password to be weak – and this is what typical password strength checkers do – but how can we tell that we have not missed some?

We argue that to be able to assess the strength of passwords, we need to understand how they were constructed. Passwords are constructed by *people*, and people follow guidelines and mental protocols when performing tasks. Therefore, a better understanding of passwords requires a better understanding of people – or at least how people construct passwords.

It is meaningful to think of passwords as strings that are composed of *components*, where components are *dictionary words*, *numbers*, and *other characters*. When producing a password, a typical user composes a password from a small number of such components, using one or more *rules*. The most common rule is *concatenation*, followed by *replacement*, *spelling mistake*, and *insertion*. Here, an example of a concatenation is producing "passbay1" from the three components "pass", "bay" and "1". Use of *L33T*[1] is a common replacement strategy, creating "s3v3nty" from "seventy" by replacement of each "e" with a "3". Misspellings, of course, may either be intentional or unintentional; resulting in passwords such as "clostrofobic". Insertion, finally, produces strings such as "Christi77na", where "77" was inserted into the name "Christina". (This was the least common type of rule among those we have surveyed, and the hardest one to automatically identify the use of, so this rule was not used in the experiment we describe herein.)

The simple insight that *people* choose passwords suggests a new approach to determining the strength of a password: One can determine the components making up the password, and the commonality of each such component; then, one could consider the mental generation rules used to combine the components and make up the password – along with the commonality of these rules being used. The strength of the password, in some sense, depends directly on the commonality of the password, which in turn depends on the commonality of its components and the password generation rules used.

[1]L33T is pronounced "leet", and is a relatively common transcription of words in which letters are replaced by other characters with some resemblance to the replaced letter.

Outline. We begin by a brief overview of the related work in section 2. We then describe how to parse passwords (section 3). In section 4, we detail the approach we took. Finally in section 5, we share our observations and the results of our analysis.

2 Related Work

Various attempts have been made to understand user attitudes towards passwords [7, 10, 13]. These attempts are based on user surveys or lab studies using relatively small sample sizes. Other studies [4, 5, 20, 22] analyze large number of passwords that have been leaked to the public, typically after large-scales attacks by online fraudsters. Some recent work has looked into users' mental model of password security. People are generally not very good at accessing the strength of their passwords [6]. However, most users do recognize the difference in security requirements between sites, and correspondingly attempt to segregate passwords amongst them [14]. A recent survey examined user attitudes towards the imposition of stringent new password requirements for user accounts [18]. While users were initially annoyed by the new requirements, most felt remembering more complicated passwords was worthwhile to improve security [3].

One of the important reasons for understanding how passwords are generated is to produce better password strength checkers, allowing these to block or flag passwords that are insufficiently secure. Most of the existing password checkers are based on naive measures like password length, resilience to brute force, dictionary based attacks, or constraints like the presence or absence of certain non-alphabetic and uppercase characters [21, 2]. Another common method used by password strength checkers is to use blacklist checks on new passwords. Blacklist checks can be performed in a constant time regardless of the size [19], thereby making large blacklist checking feasible. Schechter et al. [15] proposed a password policy in which passwords chosen by too many users are blacklisted for subsequent users.

Another important reason for understanding passwords is to provide better measures of the security they provide, in order to gauge the security exposure of various systems. One such measure of security is Shannon entropy, defined as the amount of information contained in a string [17], and used by the National Institute of Standards and Technology (NIST) to represent the strength of a password [1]. However, entropy only provides a lower bound on the expected number of guesses to find a password [12].

Another method to measure password strength is the notion of *guessability*. *Guessability* is the measure of the time needed to "guess" a given password by an efficient password cracking algorithm [9]. This is *also* a lower bound on the expected number of guesses needed to find a password, since there is no assurance that the best password cracking algorithms were used, or that these cannot be improved.

Weir et al. [20] divide a large set of existing passwords into different categories based on composition, then apply various automated cracking tools, like John The Ripper. In this way they try to examine the effectiveness of NIST'ss entropy estimates in predicting the password strength by comparing the calculated bit strengths with actual guessing difficulty based on results from password cracking tools. Narayanan et al. [13] discuss a password-cracking technique based on a Markov model, in which password guesses are made based on contextual frequency of characters.

In this paper, we decompose passwords into components and score them based on the observed frequencies of the components, and the rules that bind these together. Recently, Jakobsson and Akavipat [8] took a similar approach to score the security another type of credential, composed of a sequence of dictionary words. We apply the same techniques, along with scoring of rules, to legacy credentials – passwords, to be specific. We determine frequencies of password components by training on the large RockYou dataset [16], containing 32 millions credentials. This dataset has previously been used to understand weak passwords (see, e.g., [11]).

3 Building a Parser

The parser takes a password as input and outputs the various components and rules which have been used to construct that password – or *appear* to have been used, to be specific. Components include words, numbers, individual characters and special symbols. Some components may be overlapping, e.g., a word is made up of number of characters. Our parser has built-in rules to identify three major password creation rules: *Concatenation*, *L33T* and *misspelling.*

In order to parse an input, the parser uses an *input dictionary*. It is worth noting that language has strong influence on creation of passwords and that words usually form the core on which various other operations are applied to make the password more complex. For simplicity, only English dictionaries were used. The final dictionary obtained upon merging several specialized dictionaries contained just below 670,000 words.

Note that in many cases the passwords can be broken into components in more than one way. In such cases, the parser selects the components containing words with the *maximum coverage*. *Maximum coverage* means that the total length of the combined components, which are words, is maximized. For example, "thinput" will get parsed as "thin" and "put" rather than "t", "h" and "input". When two paths produce the same coverage, the path with the greatest probability of occurrence (as judged by the frequency of use of the rules and components) is chosen. Practically speaking, this typically corresponds to the path with the smallest number of components.

Since L33T is not a proper language and many different dialects of L33T occur on the Internet, there is no perfect algorithm to convert between English and L33T. Since the rules for such conversion vary, one can use an exhaustive approach. One can construct a table containing all mappings from a token to be replaced (number or special symbol) to a list of all common replacements for that token, as seen in various L33T to English translators. For example, the character "0" is substituted for "o" and the character "|" can be substituted for both "i" and "l". Hence, the mapping table there will be entry for "@" to be mapped to "a" and "|" to be mapped to both "i" and "l".

Identification of spelling mistakes amounts to identification of words with small Levenshtein distance to dictionary words.

4 Approach

We use large password datasets that were obtained from breaking in to dropboxes used by fraudsters; a dropbox is the file used by the fraudster to store stolen credentials, or which were somehow made public on the Internet. We used one such dataset to build a statistical model of how people generate passwords, and then used this model to score passwords from the other datasets. Each dataset has different characteristics, as will be shown, which influences the average strength of the passwords and hence, the password scores. We first describe three major characteristics of all datasets.

The first obvious characteristic is the *type of resource being protected*, i.e., what would be lost if a password is stolen. We consider only the losses as perceived by the user who produces the password, and not those potentially suffered by other users as a result of the theft – nor the losses of the organization associated with the password. The rationale is this: users are influenced by the risk they associate with theft when they select their password. For financial service providers, money would be lost. For social networking sites, the user would lose face, or be inconvenienced. For a porn site, the loss may be limited to the access to the site. In addition to this, some private information – such as user name and address – may be at stake for all of these types of sites.

A second characteristic is the *demographics of users*. We note indications that demographics affect password strength in our findings.

Finally, the *collection method* could introduce a bias in the dataset. For example, if a dataset is obtained by malware attacks, the dataset may have greater percentage of passwords of people who are not security conscious than if it was obtained by the corruption of the site that stored the passwords. Similarly, a dataset associated with phishing is likely to have a greater percentage of passwords of people who are gullible than other datasets would.

Training. In order to build a model of passwords, we used the *RockYou dataset* [16]. The RockYou dataset contains 32 million passwords. The RockYou dataset is unbaised due to its collection method – it was obtained by compromise of a website using SQL Injection. During training, we parsed the passwords from the RockYou dataset and determined the frequencies of the various rules and components.

Scoring. After training on the RockYou dataset, we parsed and scored passwords from five other datasets. The first dataset, the *Rootkit dataset*, contains 64498 passwords. It was obtained by compromise of rootkit.com. This website has forums about discussions of advanced topics in computer security. Most people registering on this website are more aware about security issues than typical users are. Second, the *Paypal dataset* contains 19053 passwords. Most of the users behind the passwords are adults. The method of collection is phishing, which introduces a bias towards people who are more gullible. Third, the *Justin Bieber dataset* contains 5091 passwords and was obtained by compromise of a fan website. Hence, there is no bias due to the collection method. However, the demographics introduces a bias as most of the users are teenagers. The fourth dataset is the *Sony dataset*, which contains 17785 passwords, and which was obtained by compromise of Sony Pictures Europe. As a result, there was no bias among passwords due to the collection method. And finally, the *Porn dataset* contains 8089 passwords, and was obtained by compromise of a pornographic website. As a result, there is no bias due to the collection method. However, there is a visible bias associated with the type of resource being protected.

5 Results

Here, we analyze the usage of various rules and components in the datasets described above.

Word components form the core of a large percentage of passwords, as can be seen in figure 1. The average number of word components per password is directly related to the average strength of the password. The Justin

Bieber dataset has the highest average number of word components per password whereas the Porn dataset has the lowest. It is not surprising that the Porn dataset is associated with a low average password security – after all, the loss of a password does not cause any great damage to a user, except if the user employes the same password elsewhere, too. However, it is a bit surprising that the Justin Bieber dataset exhibits a greater password security than the Paypal dataset – after all, the protected resources for the former are simply a user profile on a fan site. The reason may be that the Justin Bieber passwords were corrupted at the site, while the Paypal passwords were stolen from gullible users, who are likely to use lower quality passwords than more security conscious users are. It may also be due to demographic differences.

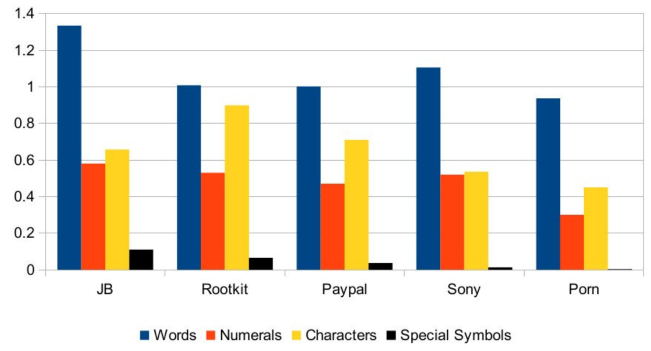

Figure 1: The figure shows the average number of components per password in the different datasets.

Average security is a bit deceptive, since it is commonly the *weakest* passwords that are guessed by cracking software. As a case in point, it can be seen that a small number of words in the Justin Bieber dataset were used to construct a much larger number of passwords than any other dataset – see figure 2 for an illustration of this. This forms the maybe best argument for why one should parse passwords: By doing that, it is possible to block the use of the most vulnerable passwords.

The remaining components include leftover characters and special symbols obtained after "cutting" words and numerals from passwords. Even though usage of special symbols in passwords increases the effective size of the character set, special symbols are unpopular among users – see figure 1.

Even in the small percentage of passwords where special symbols are used, the usage is highly skewed. Out of the 32 possible special symbols on standard ASCII keyboards, only a few of them have high frequencies. These special symbols are usually those which are used in rules like L33T e.g., "@" and "&". Most of the remaining special symbols are rarely used. In fact, we found some special symbols which were almost never

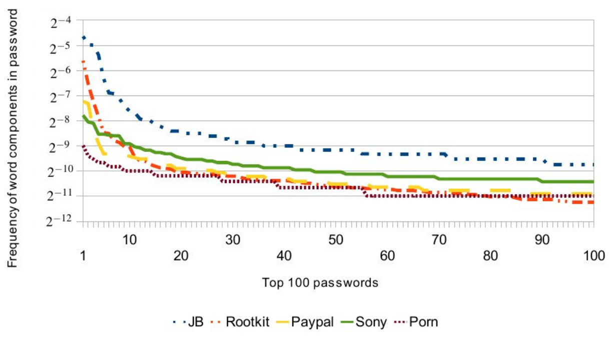

Figure 2: The figure shows the frequencies of top 100 most common words in each dataset. Even though passwords from Justin Bieber dataset usually contain a larger number of components per password, it can be clearly seen that a small number of words are reused to a greater extent in the Justin Bieber dataset, compared to other datasets.

used in any of the datasets – such as the "?" and "}". This is not a statistic anomaly: Given that the datasets contain over 100,000 passwords, this is a reminder that humans do not select passwords randomly, but rather, based on mnemonics and rules used by many others.

For a given password, the number of concatenation operations is one less than the total number of components. It can be clearly seen from figure 3 that concatenation is prevalent in all datasets. In the Justin Bieber dataset there is on average1.67 components per password, with the other datasets having lower averages. The higher usage of concatenation associated with higher security sites indicates how people typically increase password security.

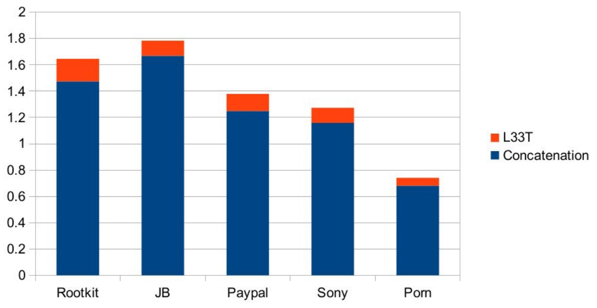

Figure 3: The average use of concatenation and L33T rules per password for different datasets.

Usage of L33T requires users to remember a new pattern replacing an old pattern which may require some effort, but can dramatically increase the strength of the password. The average number of L33T operations per password is shown in figure 3. The highest value is for Rootkits dataset. That dataset is from a corrupted website dealing related to computer security, which supports our belief that there are demographic differences between password datasets, and in particular, that "geeks" are more frequent users of L33T than others. We observe that by assigning a greater weight to the use of L33T than to the use of concatenation, we can "reward" users using uncommon rules by increasing the scores accordingly. We implicitly do this by taking the frequency of operations into consideration when determining the score of a password, and not only the frequencies of the components.

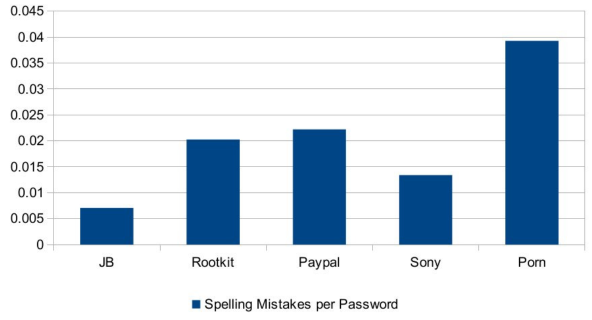

Figure 4: The average number of spelling mistakes per password for the different datasets. The use of this rule is dramatically less common than both L33T and concatenation.

The usage of spelling mistakes is quite interesting. To begin with, it is not possible for us to determine whether spelling mistakes are intentional or not, whereas the use of other rules are evidently intentional. Unintentional spelling mistakes, in the context of password strength, is therefore an excellent example of a case where "Ignorance is Bliss". Figure 4 shows the prevalence of spelling mistakes in the different datasets, showing the much greater frequency of this rule in the Porn dataset.

While the different datasets exhibit different rates of spelling errors, they are all relatively low in comparison to other rules. The low frequency of the use of spelling mistakes emphasizes the potential value of this rule to password security.

Scores. For simplicity, the score calculator we describe uses only on the frequencies of rules and components, and not the location or order of these. Therefore, we fail to take into consideration that many passwords have a "1" at the end of the password, but very few have it at another position within the password. Our score calculator is a first step towards better password strength checkers, and we believe that it can be improved considerably by further analysis along the lines of what we introduce in this paper.

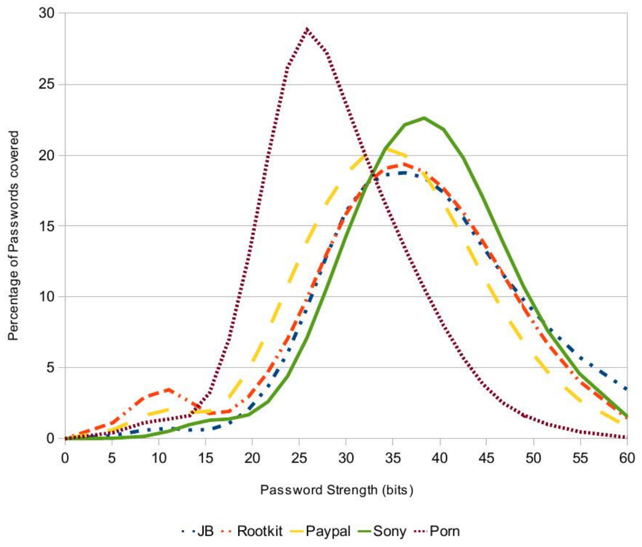

Figure 5: The figure shows the distribution of password strengths, as assessed by multiplying the frequencies of the rules and components used. The second logarithm of these values correspond to the bit strength of the passwords. It is evident that passwords from the Porn dataset are weaker than the passwords of the other datasets.

The score calculator utilizes the three different trained dictionaries: a dictionary of words, a dictionary of characters and special symbols and a dictionary of numerals. All of these are obtained by training on the RockYou dataset.

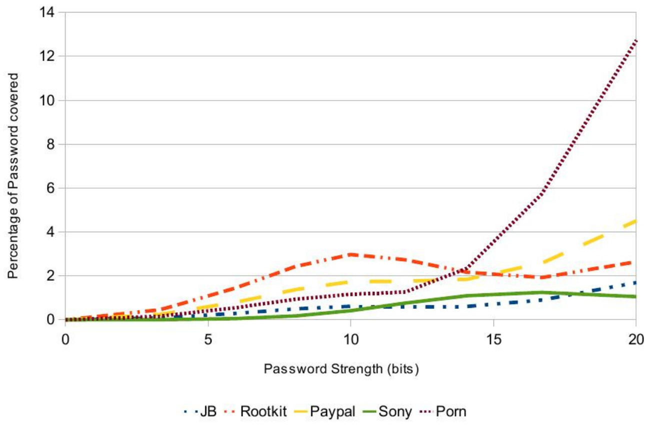

Figure 6: This figure shows a portion of the graph in figure 5. This portion of the distribution corresponds to unnecessarily weak passwords. By decomposing passwords and scoring them in the manner we describe, such passwords can be avoided.

The frequencies are the count of occurrence divided by the total count. The score calculator utilizes the frequencies of rules of occurrences of rules in that dataset. These are obtained by analysis of passwords of each particular dataset. The score of a password is simply calculated by the multiplying the frequencies of all the rules and components occurring in the password. Since all frequencies are between 0 and 1, the score will also be a (very small) value in the range of 0 and 1. The second logarithm of this product is an assessment of the bit strength of the password.

Figure 5 shows the distribution of password scores for the passwords of the different datasets. Note the close resemblance to a bell curve. Password crackers target the most frequent credentials, corresponding to the passwords with the lowest scores. The least secure region from figure 5 is magnified in figure 6.

Future Work

We anticipate that our parsing techniques can be helpful as an underlying building block to analyze and score passwords in a variety of ways, significantly improving on the result reported on herein.

There are several potentially interesting extensions to our work. We have not addressed the order of components in our scoring mechanism. Doing so will help us assign different scores to the rather common "Password1" and the not at all common "1passWord". We also have not considered scoring particular combinations of elements and rules – corresponding to creating an awareness of correlation. Our techniques can be used to compute similarity scores between passwords, e.g., to determine whether an updated password is sufficiently different from the old password, and to identify approximate reuse of credentials.

References

[1] Electronic Authentication Guideline: Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-63-1. Tech. rep., Dec. 2011.

[2] BISHOP, M., AND KLEIN, D. V. Improving system security via proactive password checking. *Computers and Security* (1995), 233–249.

[3] BONNEAU, J., AND PREIBUSCH, S. The password thicket: technical and market failures in human authentication on the web. In *Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS)* (June 2010).

[4] DELL'AMICO, M., MICHIARDI, P., AND ROUDIER, Y. Password strength: an empirical analysis. In *Proceedings of the 29th conference on Information communications* (Piscataway, NJ, USA, 2010), INFOCOM'10, IEEE Press, pp. 983–991.

[5] FLORENCIO, D., AND HERLEY, C. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* (New York, NY, USA, 2007), WWW '07, ACM, pp. 657–666.

[6] GAW, S., AND FELTEN, E. W. Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security* (New York, NY, USA, 2006), SOUPS '06, ACM, pp. 44–55.

[7] HART, D. Attitudes and practices of students towards password security. *J. Comput. Sci. Coll. 23*, 5 (May 2008), 169–174.

[8] JAKOBSSON, M., AND AKAVIPAT, R. Rethinking passwords to adapt to constrained keyboards. In *Proceedings of Mobile Security Technologies, 2012* (2012), MoST '12.

[9] KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., SHAY, R., VIDAS, T., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND LOPEZ, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. Tech. Rep. CMU-CYLAB-11-008, CyLab, Carnegie Mellon University, Aug. 2011.

[10] KOMANDURI, S., AND HUTCHINGS, D. R. Order and entropy in picture passwords. In *Proceedings of graphics interface 2008* (Toronto, Ont., Canada, Canada, 2008), GI '08, Canadian Information Processing Society, pp. 115–122.

[11] LEYDEN, J. RockYou hack reveals easy-to-crack passwords. http://www.theregister.co.uk/2010/01/21/lame_passwords_exposed_by_rockyou_hack/, 2009.

[12] MASSEY, J. L. Guessing and entropy. In *In Proceedings of the 1994 IEEE International Symposium on Information Theory* (1994), p. 204.

[13] NARAYANAN, A., AND SHMATIKOV, V. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM conference on Computer and communications security* (New York, NY, USA, 2005), CCS '05, ACM, pp. 364–372.

[14] OPPLIGER, R. Microsoft .net passport: A security analysis. *Computer 36*, 7 (July 2003), 29–35.

[15] SCHECHTER, S., HERLEY, C., AND MITZENMACHER, M. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security* (Berkeley, CA, USA, 2010), HotSec'10, USENIX Association, pp. 1–8.

[16] SCHOFIELD, J. 32.6M passwords may have been compromised in RockYou hack. http://www.guardian.co.uk/technology/blog/2009/dec/15/rockyou-hacked-passwords, 2009.

[17] SHANNON, C. E. A mathematical theory of communication. *Bell system technical journal 27* (1948).

[18] SHAY, R., KOMANDURI, S., KELLEY, P. G., LEON, P. G., MAZUREK, M. L., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Encountering stronger password requirements: user attitudes and behaviors. In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (New York, NY, USA, 2010), SOUPS '10, ACM, pp. 2:1–2:20.

[19] SPAFFORD, E. H. Opus: preventing weak password choices. *Comput. Secur. 11*, 3 (May 1992), 273–278.

[20] WEIR, M., AGGARWAL, S., COLLINS, M., AND STERN, H. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 162–175.

[21] YAN, J. J. A note on proactive password checking. In *Proceedings of the 2001 workshop on New security paradigms* (New York, NY, USA, 2001), NSPW '01, ACM, pp. 127–135.

[22] ZHANG, Y., MONROSE, F., AND REITER, M. K. The security of modern password expiration: an algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 176–186.