This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Password Strength: An Empirical Analysis

Matteo Dell'Amico, Pietro Michiardi and Yves Roudier
Eurecom
2229, Route des Crêtes
Sophia Antipolis, France
Email: {matteo.dell-amico, pietro.michiardi, yves.roudier}@eurecom.fr

***Abstract*—It is a well known fact that user-chosen passwords are somewhat predictable: by using tools such as dictionaries or probabilistic models, attackers and password recovery tools can drastically reduce the number of attempts needed to guess a password. Quite surprisingly, however, existing literature does not provide a satisfying answer to the following question: given a number of guesses, what is the probability that a state-of-the-art attacker will be able to break a password?**

To answer the former question, we compare and evaluate the effectiveness of currently known attacks using various datasets of known passwords. We find that a "diminishing returns" principle applies: in the absence of an enforced password strength policy, weak passwords are common; on the other hand, as the attack goes on, the probability that a guess will succeed decreases by orders of magnitude. Even extremely powerful attackers won't be able to guess a substantial percentage of the passwords.

The result of this work will help in evaluating the security of authentication means based on user-chosen passwords, and our methodology for estimating password strength can be used as a basis for creating more effective proactive password checkers for users and security auditing tools for administrators.

I. Introduction

Even though much has been said about their weaknesses, passwords still are – and will be in the foreseeable future – ubiquitous in authentication systems for Internet applications. They have an inherent trade-off between usability and security: while strong passwords are hard for attackers to guess, they are on the other hand also difficult for the user to remember. As Richard Smith paradoxically notes, password best practices imply that "the password must be impossible to remember and never written down" [1]. In light of this, it is not very surprising that users often knowingly choose to use weak passwords or circumvent security best practices, since they perceive that following them would get in the way of doing their work [2], [3].

To think sensibly about the security of systems that use passwords, it is therefore essential to properly evaluate their resilience to guessing attacks: this is done by comparing search space size (i.e., number of guesses) against the percentage of passwords that would be broken by such an attack. This measure does not depend on the particular nature of the authentication system nor on the attacker capabilities: it is only related to the attack technique and to the way users choose their password. The attack model and the characteristics of the system will instead define the cost that the attacker has to pay for each single guess. By combining this cost with a measure of the search space, it becomes possible to obtain a sound cost-benefit analysis for attacks based on password guessing on an authentication system.

Known Results: Current studies only provide partial answers to the question of determining the percentage of passwords that would be broken as a function of search space size: they generally focus on passwords discovered with a single kind of attack, and neglect to quantify how strong the remaining share of passwords are against more general attacks.

Analyses on dictionary attacks report a percentage of broken passwords varying between 17% and 24% [4]–[6], depending on the dataset and dictionary size. These dictionaries can be extended by systematically mangling the words; a technique proposed recently by Weir et al. manages to discover up to roughly a third of the passwords [7].

Narayanan and Shmatikov [8] proposed a technique that uses Markov chain modeling instead of naive brute force to perform guessing at a larger scale. Unfortunately, their work was only validated on a dataset of 142 hashed passwords; while 96 (67.6%) were successfully broken, nothing is known about the remaining and stronger ones.

In some studies [9], [10], the ratio of cracked passwords is reported against the running time of a specific password cracker over an hashed password file. While this metric is useful to compare implementations of password recovery programs, it falls short when trying to decouple password complexity (which is dependent on guessing strategies and user behavior) from system-specific characteristics such as hashing algorithm and computational power of the attacker. Our measurement of the search space avoids these limitations.

In a 2007 study [11], Florencio and Herley analyzed data about the passwords of about 500,000 users. That work provides interesting insights about user habits, but only quantifies password strength with a simple "bit strength" measure based on their length and on the use of uppercase, numeric, and non-alphanumeric characters; from that measure, it is basically impossible to infer resilience against advanced password-cracking techniques (e.g., [7], [8]).

Our Contribution: In this paper, we compare the search space versus number of cracked passwords for guessing techniques including dictionary attacks, brute force, dictionary mangling, probabilistic context-free grammars [7], and Markov chains [8]. We cross-validate our experiments on three large datasets of passwords, different in terms of application domain and of user localization.

In our analysis we evaluate different password cracking

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

techniques using homogeneous metrics and on the different datasets, allowing us to compare the effectiveness of each technique. We benefit from having access to the passwords in plain text, thus being able to evaluate passwords that even very powerful attackers would not be able to break. We find that different attack techniques are more effective depending on the search space size that the attacker can afford to explore, and we propose an elaborate attack model using a combined approach that adopts different guessing techniques in cascade.

Until now, the Markov chain technique has only been implemented and evaluated by taking into account the frequencies of substrings of fixed length (2 or 3) [8], [10], [12]; we find that the length of substrings is a key parameter that, when properly tuned, allows to discover some "easy" passwords with orders of magnitude of spared guesses.

We discover that password strength has an extremely wide variance: as a first approximation, the probability to guess a password at each attempt decreases roughly exponentially as the size of the explored search space grows; a surprisingly high percentage of users have extremely strong passwords that appear very difficult to guess. These diminishing returns imply that, in most cases, an attacker would eventually find a point where the cost of continuing the attack would not be justified by the probability of success. This study provides figures that can help system designers in assessing the security of their systems by evaluating where that point resides.

Our evaluation of the search space tied to a password can be straightforwardly used in real systems to create better proactive password checkers for users and security auditing tools for system administrators; coupled with information about the cost of a single guessing iteration, the cost of breaking any password can be estimated.

II. Related Work

In this section we provide a short review of studies about password security, and make the case for the importance of measuring password strength. Attacks such as phishing or social engineering, where the user is misled in communicating the password to the attacker, are unrelated to password strength and therefore outside the scope of this work.

Pricing Via Processing: To defend against intruders who repeatedly try password after password, it is possible to limit the rate at which the attacker is allowed to try new passwords by requiring the user to perform an action with a moderate cost. The following measures belong to this category:

- CAPTCHAs [13], which require solving puzzles that are difficult without human intervention;
- key strengthening techniques, which require a few seconds of computation to derive a key from the passwords.

It is noteworthy that these techniques impose a trade-off to legitimate users: if an honest user has to pay a cost c, the attacker must pay at most $c \cdot s$, where s is the strength of the password in terms of the number of attempts needed to guess it. The measures obtained in this paper can be used to estimate costs and benefits of these systems, and thus to properly tune this c parameter.

An alternative approach blocks accounts after a given number of failed attempts. This response, however, opens the door to denial of service attacks on user accounts and is ineffective against attacks not targeted towards a single user [14].

Offline Attacks: In most cases, the authentication server does not store passwords in plain text. Instead, it keeps an "encrypted" version of them which is conceptually analogous to a hash: when a user attempts to log on, the password they provide is hashed and compared to the stored value. In this way, even if an attacker obtains the hashed passwords, they cannot be used right away to log on to the system. To make it costly for the attacker to guess the password by hashing lots of password candidates, key strengthening techniques are again applied. Attacks based on pre-computing the hashed version of the most likely passwords [8], [15] are defeated with "salting": appending a random number to the password before hashing it, and then storing this number along with the hashed password.

Since these techniques are based on the idea of making guessing attacks costly, the password strength that we are measuring is also a key parameter when evaluating the resilience of a password system to offline attacks.

Proactive Password Checking: A proactive password checker is a system that forces (or advises) the user to choose complex enough passwords. The impact of these checkers on actual password security is debatable: as Wu [16] notes, "[users are] very good at selecting passwords that are just 'good enough' to pass whatever checking is in place". See, for example, the discussion on MySpace passwords in Section III. Furthermore, a proactive password checker could encourage users to use non-dictionary passwords that are related to their personal life such as dates, telephone numbers or license plate numbers [2]. For a motivated attacker, these passwords are even easier to guess than dictionary words.

Existing password checkers are based on quite naive metrics [17], [18]: they are generally based on password length, resilience to "brute force", dictionary based attacks and/or heuristics based on presence of non-alphabetic and uppercase characters; still, they do not take into account advanced cracking techniques. Our measure of strength as search space size can be used as the basis for more effective password checkers.

III. Datasets

IT: The "Italian" dataset: This dataset contains the unencrypted passwords for the registered users of an Italian instant messaging server adopting the XMPP protocol[1], administered by one of the authors. Our analysis only discloses aggregate information about the passwords of users, and actually constitutes part of a security audit of the system. Storing passwords in plain text on the server is required by authentication algorithms such as CRAM-MD5[2]. User registration is free (any unused username may be taken) and no policy for password strength is enforced.

[1] http://xmpp.org/

[2] http://tools.ietf.org/html/rfc2195

Dataset	Size	#unique	Avg. length	#characters
IT	9,317	7,848	7.86	124
FI	15,812	13,395	7.60	90
MS	33,671	30,690	8.10	96

TABLE I
SUMMARY INFORMATION ABOUT DATASETS: NUMBER OF USERS, OF UNIQUE PASSWORDS, AVERAGE PASSWORD LENGTH AND TOTAL NUMBER OF CHARACTERS IN ALL PASSWORDS.

FI: The "Finnish" dataset: This dataset comes from a list of passwords that were publicly disclosed in October 2007 by an unknown group[3]. The list contained both hashed (MD5) and unencrypted passwords, mostly from different Finnish web forums. We limited our analysis to the unencrypted disclosed passwords, all from the same website.

MS: The MySpace dataset: These passwords were obtained through a phishing attack on a fake MySpace login page, and were disclosed in October 2006 [19], [20]. Usernames, in this case, are email addresses. While this is the largest dataset we are analyzing, there are some shortcomings with it: first, we only have the passwords of less security-conscious users who fell for the attack; second, users may have (on purpose or inadvertently) put wrong passwords on their phishing page. MySpace requires users to insert both alphabetic and non-alphabetic characters in their passwords; this imposes an artificial impact on passwords that users, left alone, would choose. By analyzing the differences between this dataset and the former ones, we can estimate the effect of this requirement on password strength.

Dealing With Dataset Quality: Since the FI and MS datasets come from lists of publicly disclosed passwords, we cannot be completely confident that they are an accurate representation of the users' actual passwords. On the other hand, we are confident in the quality of the IT dataset: it contains the passwords of all registered users, and we know the policies enforced on the server. For this reason, we will base our analysis on IT, turning onto the other datasets to confirm that our results generalize beyond a single set of passwords.

Our datasets reflect the common case where users are free to register on a network service, and use it to establish a persistent identity. The threat to the user is that attackers stealing their passwords would be able to impersonate them, perhaps to harm the reputation of the attacked user, to exploit the trust obtained from other users or to gain access to sensitive information. The same kind of threat would apply to any system that uses authentication to establish the origin of communications between users such as, for example, e-mail.

A First Look At the Datasets: Table I summarizes some information about our datasets. It is interesting to note that in all cases some users share the same password. This may be due to coincidences and use of too frequent passwords, but this may be also caused by the same people registering under different usernames at the same server. The average password length is close to 8 in all cases, and the number of used characters is higher in IT because arbitrary Unicode characters are allowed, and used sparingly by the users.

Expression	Example	IT	FI	MS
[a-z]+	abcdef	51.21%	53.06%	1.09%
[A-Z]+	ABCDEF	0.29%	0.17%	0%
[A-Za-z]+	AbCdEf	53.74%	54.04%	1.09%
[0-9]+	123456	9.10%	3.43%	0.15%
[a-zA-Z0-9]+	A1b2C3	93.43%	95.43%	90.43%
[a-z]+[0-9]+	abc123	14.51%	27.10%	77.39%
[a-z]+1	abcde1	0.26%	1.43%	19.89%
[a-zA-Z]+[0-9]+	aBc123	16.30%	28.03%	77.48%
[0-9]+[a-zA-Z]+	123aBc	1.80%	2.16%	5.76%
[0-9]+[a-z]+	123abc	1.65%	2.09%	5.75%

TABLE II
PERCENTAGE OF PASSWORDS MATCHING VARIOUS REGULAR EXPRESSIONS.

In Table II we compare the matching ratio of different regular expressions in our datasets. In all cases, non-alphanumeric characters are present only in less than 10% of the passwords. It is very interesting to compare the matching ratios of IT (with no strength enforcement measures) with MS (where a mixture of alphabetic and non-alphabetic characters is required). In MS, a small number of all-alphabetic or all-numeric passwords are present, and this may be due to users inadvertently or knowingly inserting wrong passwords in the phishing page.

As already noticed by Sebastian Porst [20], most MySpace users comply with the requirement of inserting a non-alphabetic character by appending a number at the end of the password – roughly 20% of the users actually comply by adding a "1". The impact of this measure on password strength may appear therefore quite debatable, especially in the case that the attacker knows about the requirement.

Some users in IT appear to have a stronger tendency towards choosing stronger passwords with less easily detectable structure: as we will show in the following, the complex passwords from that dataset are the most difficult ones to break.

IV. DICTIONARY ATTACK

Dictionary attack is the most effective technique to guess the weakest passwords. We adopted the dictionaries available in the well known *John the Ripper* (JtR) password recovery tool. The extended dictionaries that we used are available for paid download from the program website[4].

The Dictionaries: The JtR dictionaries contain words from 21 different human languages, plus a list of frequently used passwords. For some languages (like English and Italian), various dictionaries of different sizes are available: the smaller ones contain only the most frequently used words while the bigger ones also contain more obscure words, the rationale being that more common words are more likely to be chosen as passwords. Taken together, all dictionaries account for almost 4 million words.

A known technique to create strong but easy to remember passwords is to turn phrases into passwords by extracting an acronym, possibly also using punctuation. For example, the phrase "Alas, poor Yorick! I knew him, Horatio" becomes "A,pY!Ikh,H". We also evaluated such acronyms with a dictionary created by Kuo et al. [21] that was put together

[3] http://www.f-secure.com/weblog/archives/00001293.html

[4] http://www.openwall.com/wordlists/

Dictionary (size)	IT		FI	
	Found	Guess pr.	Found	Guess pr.
Frequent (2.8K)	5.95%	$2.1 \cdot 10^{-5}$	2.86%	$1.0 \cdot 10^{-5}$
English 1 lc (27K)	4.91%	$1.8 \cdot 10^{-6}$	3.38%	$1.2 \cdot 10^{-6}$
English 2 lc (297K)	9.42%	$3.2 \cdot 10^{-7}$	6.26%	$2.1 \cdot 10^{-7}$
English 3 lc (390K)	11.59%	$3.0 \cdot 10^{-7}$	7.53%	$1.9 \cdot 10^{-7}$
Extra lc (445K)	8.03%	$1.8 \cdot 10^{-7}$	8.16%	$1.8 \cdot 10^{-7}$
Italian 1 lc (63K)	3.71%	$5.9 \cdot 10^{-7}$	0.79%	$1.3 \cdot 10^{-7}$
Italian 2 lc (344K)	14.89%	$4.3 \cdot 10^{-7}$	6.62%	$1.9 \cdot 10^{-7}$
Finnish lc (359K)	8.45%	$2.4 \cdot 10^{-7}$	20.24%	$5.6 \cdot 10^{-7}$
All above (1.45M)	24.79%	$1.7 \cdot 10^{-7}$	26.02%	$1.8 \cdot 10^{-7}$
All JtR dicts (3.9M)	25.94%	$6.6 \cdot 10^{-8}$	26.97%	$6.6 \cdot 10^{-8}$
Mnemonics (406K)	1.27%	$3.1 \cdot 10^{-8}$	0.35%	$8.7 \cdot 10^{-9}$

TABLE III
DICTIONARY ATTACKS.

by scraping websites displaying memorable phrases, such as citations and music lyrics.

Experimental Results: We simulated dictionary attacks with all the JtR dictionaries on the IT and FI datasets. The rule requiring non-alphabetical characters makes basic dictionary attacks essentially pointless on the MySpace passwords. Table III shows the results for the most representative instances.

The "lc" acronym stands for all-lowercase dictionaries: those containing uppercase letters are matched by very few words in our dataset. The English 1, English 2 and English 3 dictionaries, like Italian 1 and Italian 2, are listed in growing size; each word belonging to a smaller dictionary is also contained in the bigger versions. The "Extra" dictionary contains likely passwords such as proper nouns, misspellings or alterations of words. The "found" column lists the percentage of passwords appearing in that dictionary; the "guess probability" column reflects the probability that a random word from that dictionary matches a random password; it can be computed by dividing the ratio of found passwords by the dictionary size. A rational attacker would try a word from that dictionary only if the benefit of cracking the password exceeds the inverse of that probability times the cost of the effort for trying that password.

The dictionaries have non-empty intersections, corresponding to words that are quite common. This explains why Italian users seem keen on choosing Finnish words as passwords and vice versa, and why the guessing probability in the "all above" row is lower than for each of the contained dictionaries: those repeated words are counted only once in the union dictionary.

An interesting feature is the noticeably higher density of common English words (those present in the small "English 1" dictionary); that phenomenon is much less relevant with respect to Italian in the IT dataset (unfortunately, we don't have a dictionary of common Finnish words to confirm this finding on FI). We think that this is caused by the fact that most users know English as a second language, and thus are less inclined to use an obscure word as their password. This suggests that basing the password on one's native language could be a good advice to increase password strength without requiring significant additional effort.

The most important lesson drawn from this data is the principle of *diminishing returns*: the probability of guessing a word sharply decreases as the dictionary grows. A small dictionary of 2,800 frequent passwords cracks 6% and 3% of the passwords respectively in IT and FI; with a 500-fold increase in the size of the dictionary up to almost 1.5 million, the number of cracked passwords rises to 25% and 26%. By increasing again the dictionary size by a factor of 2.7 (including other language dictionaries), only 1% more passwords are discovered. To put it in another way, the probability of guessing a given password by trying an element of the "frequent passwords" dictionary is one in 47,000 in IT and one in 99,000 in FI. On the other hand, after having tried all the frequent passwords, the Italian, Finnish, and English dictionary, the probability of guessing by using another dictionary word is less than one in 200 million! Since the guessing probability decreases so sharply, it is conceivable that in many cases it won't be worth trying a bigger dictionary for the attacker.

We also observe that the mnemonic dictionary is quite ineffective. This may be due to several reasons: first, few users actually use mnemonics for their passwords; second, they are actually much harder to break with dictionary attacks. Moreover, we are not able to ascertain whether the habit of choosing English passwords for Italian and Finnish users would carry over to the use of mnemonics. Our data is, at the moment, insufficient to point towards one of these reasons.

V. MANGLING

Many users adopt simple techniques to protect passwords against dictionary attacks. Some examples are juxtaposition of words, appending or prepending sequences of digits or symbols to passwords, or capitalizing words. The technique of mangling is directed towards this goal: new candidate passwords are generated by rules altering dictionary words. *John the Ripper* can use mangling rules to generate extended set of passwords; we applied them to the "all dictionaries" list (3.9 million elements) to generate a mangled list of 147,945,837 candidate passwords. With the extended dictionaries described in the previous section, JtR also ships a hand-tuned dictionary containing 40,532,676 candidates – mangling rules are chosen depending on the dictionary, with a different number of rules applied to each dictionary. This smaller dictionary is not a proper subset of the first, and contains some words that cannot be generated using the default rules of JtR.

Probabilistic Context-Free Grammars: Recently, Weir et al. proposed a new technique for dictionary mangling based on probabilistic context-free grammars (PCFGs) [7]. According to this technique, a probabilistic model is obtained from a training set of clear-text passwords, in two steps. First, the "structure" of the password is obtained and mapped to a context-free grammar production: for example, the "$abc123" password maps to the $S \rightarrow \mathbf{S_1L_3D_3}$ production (S is the starting non-terminal), representing a sequence of one symbol, three letters, and three digits; the production is assigned a probability equivalent to its frequency in the training set. The L_i productions are created based on the words from the dictionary to be mangled, while the S_i and D_i productions are obtained, again, from the training set: for example, if the $\mathbf{D_3} \rightarrow 123$ production is assigned a probability 0.4, this means that 40% of all sequences of three digits in the dataset correspond to the string "123".

This technique makes it possible to create a set of candidate passwords, and to assign a probability to each one of them. In their work, Weir et al. designed an efficient algorithm to return an arbitrary number of productions by decreasing order of probability (details can be found in [7]).

Experimental Setup: We created a training set from each of our datasets, randomly choosing half of the passwords in each of them. We then used each training set to create three PCFG dictionaries mangling the "all languages" dictionary of JtR, with different sizes. To allow easy comparison with dictionary attack and the two JtR mangled dictionaries, we selected the following sizes: 1.45 million, to match the "all above" line in Table III; 40.5 millions and 147.9 millions to match the JtR dictionaries. We then simulated a dictionary attack using the nine dictionaries generated, plus the two JtR dictionaries, against our three datasets.

When evaluating a PCFG dictionary against the dataset from which we obtained the training set, we only used the half of the passwords that was not part of the training set.

Since the MySpace passwords must contain alphabetic and non-alphabetic characters, it is pointless for an attacker to use candidates that don't satisfy this requisite. We therefore considered an attack where those passwords were filtered out from the mangled dictionary. A small number of additional passwords are found when the algorithm is run using the non-filtered dictionary: this is due to the passwords in MS that do not conform to the security requirements of MySpace, as discussed in Section III.

Results: The results of our experiments, using the same format we used for standard dictionary attacks previously, are reported in Table IV. Our major conclusions are:

1) The principle of diminishing returns continues to apply: as the dictionary size grows, the probability that the password will be found decreases with each single guess.
2) The strength enforcement policy applied by MySpace appears to pay off only if the attack does not expand to include mangled dictionaries: with a size of 1.45 million candidates, the passwords in the MS dataset appear stronger; this advantage is lost when the mangled dictionaries reach the size of 41 million.
3) PCFGs prove themselves very useful for the search space range under scrutiny: they perform better than the automatic mangling rules applied by John the Ripper, and they are comparable to the hand-tuned mangled dictionary. Wise attackers would not however use PCFGs before relevant dictionaries, since the latter ones are more likely to find the correct password early on.

The passwords in IT are more complex to break using these techniques, reflecting a difference in user behavior when choosing the password.

Using IT as a training set for FI, and vice versa, appears quite effective, despite of the difference in user language and application. On the other hand, MS is a poor training set for both IT and FI; this is easily explained by the password strength enforcement rules.

To evaluate the limits of the PCFG approach, we verified the percentage of passwords that would be found if the PCFG generator would be left running indefinitely, as described in the following. A password will obviously never be generated if the corresponding productions don't exist in the grammar (because the training set does not contain passwords with the same structure or matching sequences of symbols or digits, or because the password contains a sequence of letters not appearing in the starting dictionary). When taking into account the matching training set, it is possible to break 60.95% of the MySpace passwords (the grammar can produce a total of $4.52 \cdot 10^{18}$ guesses), 52.30% of the "Finnish" passwords (totaling $2.00 \cdot 10^{26}$ guesses), and 44.17% of the "Italian" passwords ($3.87 \cdot 10^{155}$ guesses). The huge variations in the number of guesses are due to few complex passwords: most candidates coming from the Italian training set are due to a single 130-character long password. All these guesses would however be labeled with a very low probability, and will therefore never be generated in a realistic attack.

VI. Markov Chain-Based Attack

When even mangled dictionaries are unsuccessful, attackers don't need to resort to an exhaustive brute-force attack: some passwords are much more likely to be chosen than others, even when they are not based on dictionary words. Various regularities exist: passwords are usually made of pronounceable sub-strings and/or sequences of keys that are close on the keyboard. State-of-the-art password retrieval tools such as John the Ripper [12] and AccessData's Password Recovery Toolkit [22] employ Markov chains to narrow the search space that would need to be explored with brute force.

In this section, we describe and validate an attack based on Markov chain-based modeling of the frequencies of substrings with parametric length k, or k-graphs. This allows us to label candidate passwords with variable probabilities, where strings that are labeled as more likely are checked first. Some password generating utilities actually use this kind of modeling to obtain meaningless but pronounceable passwords on the grounds that they're easier to remember, thus sacrificing some strength for usability[5].

A. The Technique

We adopt the techniques introduced by Narayanan and Shmatikov [8], applying the model also to substrings of length 3 and more. This model associates each password with a probability p, representing a password choice as a sequence of random events: first, the length of the password is chosen according to a given probability distribution; then, each character of the string gets extracted according to a conditional probability depending on the previous $k-1$

[5] See for example gpw (http://www.multicians.org/thvv/tvvtools.html#gpw), apg (http://www.adel.nursat.kz/apg/), otp (http://www.fourmilab.ch/onetime/).

Dictionary	IT		FI		MS (no filter)		MS (filtered dictionary)		
(training set) (size)	Found	Guess pr.	Found	Guess pr.	Found	Guess pr.	Found	Guess pr.	Filtered dict size
PCFG (IT) (1.45M)	24.64%	$1.7 \cdot 10^{-7}$	24.35%	$1.7 \cdot 10^{-7}$	0.90%	$6.2 \cdot 10^{-9}$	0.21%	$1.3 \cdot 10^{-7}$	17,015
PCFG (FI) (1.45M)	23.47%	$1.6 \cdot 10^{-7}$	24.43%	$1.7 \cdot 10^{-7}$	0.75%	$5.2 \cdot 10^{-9}$	0.06%	$6.9 \cdot 10^{-8}$	9,413
PCFG (MS) (1.45M)	2.14%	$1.5 \cdot 10^{-8}$	2.44%	$1.7 \cdot 10^{-8}$	13.02%	$9.0 \cdot 10^{-8}$	12.98%	$9.0 \cdot 10^{-8}$	1,447,290
JtR hand-tuned (41M)	30.11%	$7.4 \cdot 10^{-9}$	31.29%	$7.7 \cdot 10^{-9}$	31.77%	$7.8 \cdot 10^{-9}$	31.02%	$1.0 \cdot 10^{-8}$	30,258,334
PCFG (IT) (41M)	30.88%	$7.6 \cdot 10^{-9}$	36.17%	$8.9 \cdot 10^{-9}$	30.93%	$7.6 \cdot 10^{-9}$	30.22%	$8.1 \cdot 10^{-9}$	37,114,836
PCFG (FI) (41M)	29.53%	$7.3 \cdot 10^{-9}$	41.13%	$1.0 \cdot 10^{-8}$	32.88%	$8.1 \cdot 10^{-9}$	32.16%	$8.8 \cdot 10^{-9}$	36,709,144
PCFG (MS) (41M)	20.88%	$5.2 \cdot 10^{-9}$	28.97%	$7.1 \cdot 10^{-9}$	38.52%	$9.5 \cdot 10^{-9}$	37.88%	$9.5 \cdot 10^{-9}$	39,674,064
JtR mangled (148M)	29.56%	$2.0 \cdot 10^{-9}$	31.53%	$2.1 \cdot 10^{-9}$	24.16%	$1.6 \cdot 10^{-9}$	23.41%	$2.2 \cdot 10^{-9}$	105,029,406
PCFG (IT) (148M)	33.12%	$2.2 \cdot 10^{-9}$	41.81%	$2.8 \cdot 10^{-9}$	43.62%	$2.9 \cdot 10^{-9}$	42.90%	$3.0 \cdot 10^{-9}$	144,323,223
PCFG (FI) (148M)	31.52%	$2.1 \cdot 10^{-9}$	44.21%	$3.0 \cdot 10^{-9}$	42.14%	$2.8 \cdot 10^{-9}$	41.41%	$2.9 \cdot 10^{-9}$	140,673,878
PCFG (MS) (148M)	30.28%	$2.0 \cdot 10^{-9}$	41.18%	$2.8 \cdot 10^{-9}$	48.27%	$3.3 \cdot 10^{-9}$	47.46%	$3.3 \cdot 10^{-9}$	145,480,767

TABLE IV
DICTIONARY ATTACKS WITH MANGLING TECHNIQUES AND PROBABILISTIC CONTEXT-FREE GRAMMARS (PCFGS).

characters. These probabilities are obtained by observing how often, on a suitable training set, these $k-1$ characters are followed by the same k-th one.

In the effort of creating an algorithm to generate password guesses sorted by descending probabilities, Narayanan and Shmatikov also described an efficient recursive algorithm to approximate the number of passwords with a probability of occurrence in the model higher than or equal to p. For a formal description of the model and details on the algorithms, we refer to the original paper [8].

In the absence of a representative training set of plaintext passwords, a dictionary can be used. As we will experimentally show, using passwords from the same dataset as training set finally results in a noticeably better model. In this case, when considering a given password in our experiments, that password itself is removed from the training set and is not taken into account when computing its corresponding probability p.

As mentioned in Section III, some users share the same password. This might be due to chance and to the fact that those passwords are quite trivial; another possibility is that they come from the same user registering many accounts with the same passwords. In the latter case, an attacker would not have access to the password in a representative training set, and it would be correct for our purposes to remove all copies of the password from the training set. Since we cannot discriminate between the two cases, we will adopt a conservative approach that may result in overestimating the capabilities of the attacker, therefore discarding only a single copy of the password from the training set.

A model with higher values of k should be more accurate, but the process of creating it is more difficult and expensive. In the extreme, a model with k exceeding the maximum password length would explicitly list the probability of occurrence of each possible password: this would require prohibitive training set size and storage capabilities (the required space is of the order of $|C|^k$, where $|C|$ is the size of the character set). With limited resources, when a k-graph does not appear in the training set due to under-sampling, then the probability of a password containing that k-graph is computed as 0. Such a model would therefore never generate the required password.

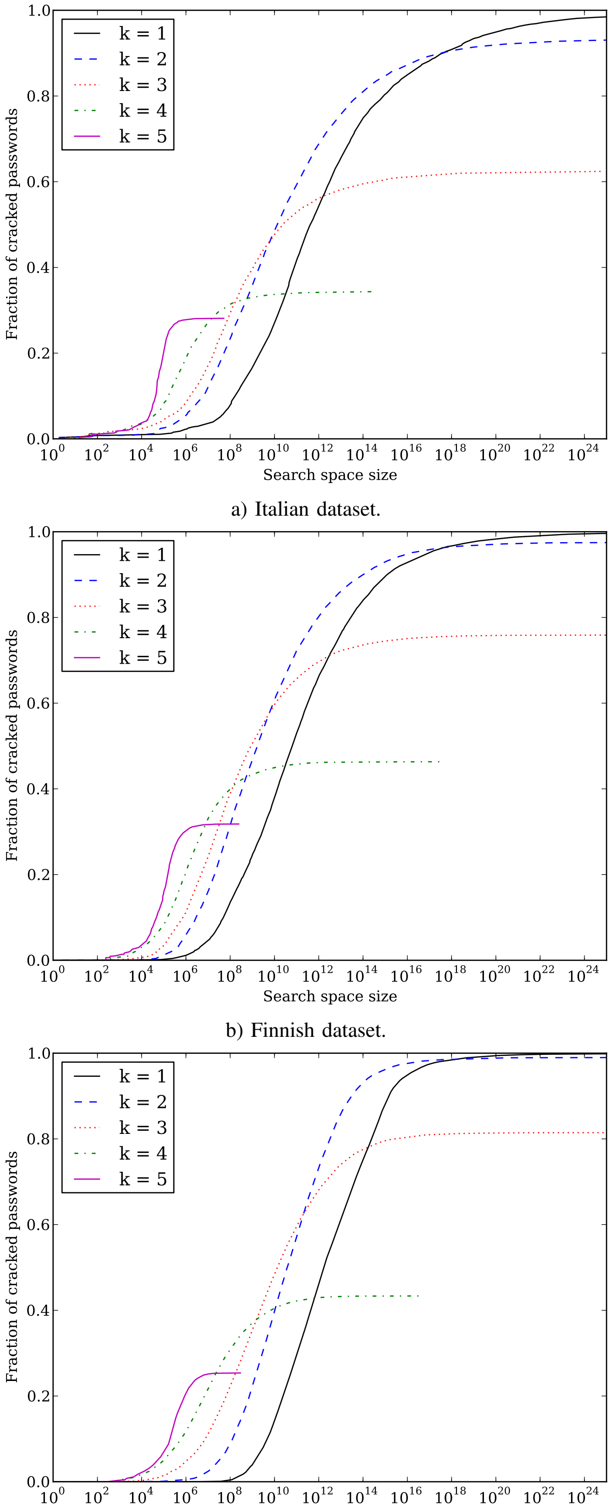
B. *Experimental results*

This section describes the results of the experiments described above when applied to our password datasets. Unless otherwise specified, we use the passwords from the same dataset as training set.

We use the approximated algorithm described in section VI-A to compute the search space needed to break a password once its corresponding probability p is computed, when the search space is too big to be generated explicitly; since we aim for a conservative estimate that approximates by excess the capabilities of the attacker, we implemented the algorithm so that it would always return an under-estimation of the search space size. Our experiments with this approximated technique (not shown due to lack of space) result in a relative error of the order of 5%.

Password Strength: In Figure 1, we plot the fraction of passwords guessed as a function of the search space size in our three datasets. In all cases, the results are qualitatively very similar. With higher values of k, we obtain better results for the weaker passwords due to the more precise modeling obtained in this case. However, the passwords that include k-graphs not represented in the training set cannot be guessed. Methods based on smaller k values become more effective because they can "generalize" some more. In practice, the optimal strategy depends on the resources of the attacker, measured by the number of attempts that can be tried. It should be noticed, anyway, that dictionary attacks and mangling techniques produce better guesses when the search space has a size below roughly 10^8.

The "diminishing returns" effect also applies to this technique: when choosing the best value of k for each case, taking into account the IT dataset, around 100,000 candidates need to be tried in order to guess 20% of the passwords ($k=5$); this number rises to roughly 1.1 billion candidates for a success rate of 40% ($k=3$); the search space needed to break 90% of the passwords grows to approximately $3 \cdot 10^{17}$ ($k=2$). Similar order of magnitude variations are observable also in this case. With such a huge variance in the size of the search space, it seems that no reasonable attack based on password guessing would succeed in guessing all passwords – excepting those cases where users are artificially forced to limit password



a) Italian dataset.

b) Finnish dataset.

c) MySpace dataset.

Fig. 1. Search space size versus fraction of guessed passwords.

strength, for example by imposing a maximum length.

There are noticeable differences in terms of search space size between the datasets. For MySpace, the search space for weak passwords is bigger, while it is smaller for stronger passwords. We think that this is mainly due to the particularities of the dataset: weak passwords are made stronger by the requirement of non-alphabetic characters; strong passwords created by security-conscious users, on the other hand, are under-represented since such users are less likely to fall victim to a phishing attack.

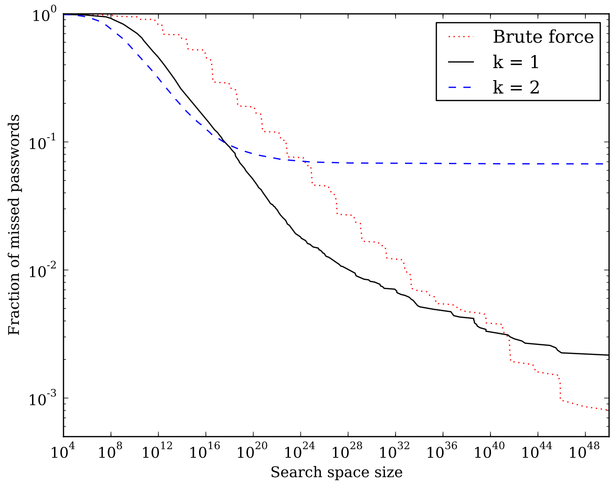


Fig. 2. Brute force and Markov-model based attacks. Dataset: IT.

Passwords in IT appear stronger than those in FI. This confirms the remarks about lower predictability in password structure that we highlighted in the previous sections.

Brute Force: In Figure 2, we compare brute force with the Markovian modeling on IT. The brute force approach starts with the empty password, then proceeds with enumerating all possible passwords with increasing length. The full Unicode character set currently has more than 99,000 characters[6], but many of them are rare and definitely unlikely in a password; to account for this, we again took a conservative approach overestimating the attacker capabilities, and took into account only the 124 characters that we have found in our dataset.

In all but the most extreme cases, the Markovian model proves more efficient by orders of magnitude. It is not before 10^{40} candidates (and having found 99.7% of the passwords) that a brute force approach becomes more effective than the Markovian model with $k = 1$ (character frequencies). This number is well beyond the capabilities of any realistic attacker: to put this in context, a cluster of a thousand 10 GHz machines would need more than $3 \cdot 10^{19}$ years to reach that number of iterations, even assuming that they are able to try a password for each clock cycle.

Training Sets: Figure 3 shows how the choice of training set affects attack performance on the IT dataset. The "common passwords" dictionary from JtR is more representative of real passwords than standard dictionaries, since it contains combinations of characters, such as punctuation and digits. Still, it appears that "average" passwords do not closely resemble the most common ones.

[6] http://www.unicode.org/press/pr-ucd5.0.html

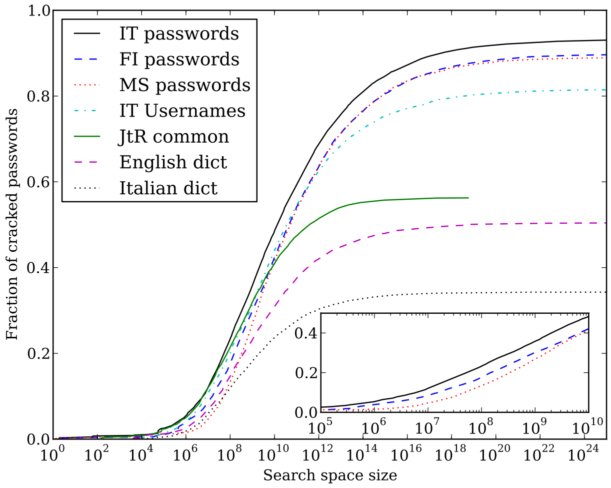


Fig. 3. Using different training sets to guess passwords in IT $(k = 2)$. In the inner frame: detail on the FI, IT and MS password training sets.

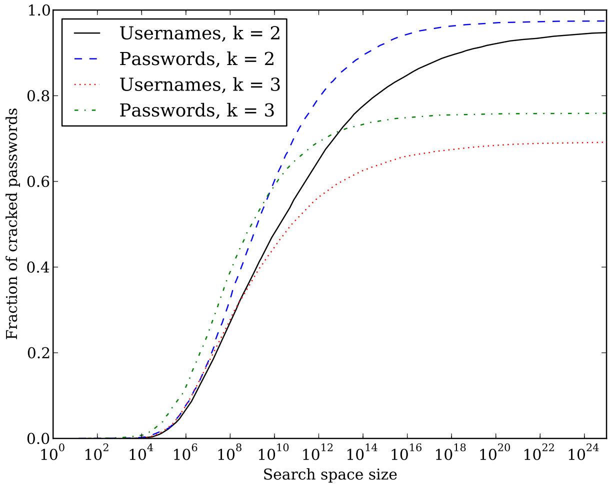


Fig. 4. Comparison of complexity between passwords and usernames in IT.

Password datasets are the most effective training sets; unsurprisingly, the most effective training set for a password in IT is the set of remaining passwords in the same dataset. By using passwords in FI, not much is lost; we attribute the variation mainly to the difference in language. The case of MS passwords as training set is interesting: they are basically analogous to FI for strong passwords, but they do not represent weak ones well. We believe this is due to the over-representation of the required non-alphabetic characters in MySpace passwords.

If a representative training set of real passwords is not available to the attacker, usernames are by far the most effective training set. It appears that, when users are asked to provide a username and a password, they employ similar criteria. This is quite surprising since the two strings need to satisfy very different, and arguably conflicting, criteria: good usernames are easily memorable, while a strong password has to be as difficult to guess as possible.

Usernames: The former result suggests a consideration: usernames and passwords are chosen simultaneously, when registering a new account. A user wants both strings to be memorable, since the two are needed in order to log on successfully. However, while there is no incentive in choosing complex usernames[7], a security conscious user will commit some effort to make his password more complex.

The difference in complexity between usernames and passwords is therefore a way to measure the effort that users willingly put in making their passwords more complex: while usernames can be very long or difficult to guess, this is not likely to happen as the result of a conscious attempt to do so.

In Figure 4, we compare the search space size associated to usernames and passwords. Matching what we have done with passwords, the training set used to guess a given username consists of all the usernames except the one under scrutiny. It turns out that the effort that users put in creating complex passwords is measurable, but it is overall quite weak: given a choice for k and a search space size, the percentage of “cracked” usernames never exceeds the cracked passwords by more than 15%.

VII. Combining Strategies

Our results confirm that no single strategy or technique is more effective independently of the search space: dictionaries are most effective in discovering the weakest passwords; the coverage (fraction of passwords that are in the dictionary) grows as the dictionary size grows, but this entails a loss in precision (fraction of dictionary items that are actual passwords). Mangling is effective when dictionaries are exhausted, but it cannot be used to guess all passwords either. The Markov-chain based technique should be used if the password search space becomes very large; with this strategy, higher values of k obtain better results at first, but after a number of attempts they become quite ineffective and one needs to switch to lower values of k.

Consistently with our approach of overestimating the capabilities of the attacker in the face of uncertainty, we assume that the attacker has access to a password training set which is as effective as the one we obtain from the clear text passwords. Furthermore, we also assume that the attacker is able to predict the effectiveness of techniques that we measured in Sections IV, V, and VI. For reasons of space, we limit this analysis to the IT dataset.

In Table V, we summarize the cumulative explored search space size and percentage of cracked passwords after each step. Candidates that would appear in more than one step are counted only once. For the Markov chains with $k \leq 3$, the search space size has been estimated with the approximated algorithm of [8]. Since the PCFGs are generated using the training set from IT, we only consider the passwords that are not part of that training set.

These results constitute the answer to our original question: how many attempts would an attacker need in order to guess

[7] Users, however, are forced to choose a second (probably more complex) username if the first one they choose is already registered.

Step	#attempts	Cracked
Common passwords	2,820	5.95%
Training set	10,143	26.20%
English 1 lc	36,694	28.00%
Italian 1 lc	98,606	29.77%
Italian 2 lc	373,923	34.20%
English extra lc	775,574	36.83%
English 2 lc	1,034,389	37.70%
English 3 lc	1,124,012	38.26%
PCFG (IT) (1.45M)	2,570,596	41.50%
PCFG (IT) (41M)	41,648,625	46.33%
PCFG (IT) (148M)	149,052,498	49.36%
Markov chain - $k = 5$	149,053,078	53.49%
Markov chain - $k = 4$	155,855,686	54.58%
Markov chain - $k = 3$	~850,000,000	61.90%
Markov chain - $k = 2$	~$7 \cdot 10^{16}$	91.44%
Markov chain - $k = 1$	~10^{40}	99.70%

TABLE V
CUMULATIVE NUMBER OF ATTEMPTS AND OF GUESSED PASSWORDS FOR THE MULTI-STEP APPROACH.

a given percentage of the passwords? By integrating this with system-specific knowledge such as the computational cost needed to perform a single guess and the amount of resources that the attacker has access to, it is possible to estimate the percentage of passwords that are vulnerable to a given attack.

VIII. CONCLUSION

In this work we focused on the empirical study of real-world passwords from three datasets, different in terms of both application domain and user localization. We implemented and used a variety of state-of-the-art techniques for password guessing, including dictionary attacks, mangling using dictionaries and probabilistic context free grammars, and Markov chain-based strategies. We proposed a unique and comprehensive analysis of the password strength of Internet applications.

We measured the resilience of passwords in terms of the search space required for an attacker to guess a fraction of the passwords contained in our dataset and we studied the properties of the different attack techniques we implemented. Our results revealed that no single attack strategy prevails over the others: dictionary attacks are most effective in discovering weak passwords; dictionary mangling is useful when the base dictionaries are exhausted; Markov-chain techniques are powerful in breaking strong passwords.

All the attack techniques that we analyzed are affected by diminishing returns: the probability to guess a password at each attempt decreases roughly exponentially as the size of the explored search space grows. Thus, the probability of success, at some point, will not justify anymore the cost for an attacker. Our results can help find this point.

Our results also shed light on some aspects of user practices in choosing their passwords: we found that, within our datasets, users put relatively little effort in choosing their password when compared to the choice of their usernames. As illustrated by MySpace, adopting restrictive password policies does not necessarily prevent the creation of weak passwords.

We believe that proactive password checkers are a better approach, and we are currently implementing one such tool based on the findings of this work: given one or more attack models such as the one described in Section VII, it will compute in real time an approximation of the number of guesses needed to crack the password. This information will be provided to the user as an approximation of password strength.

Our future research agenda will also focus on user behavior based on data we are currently collecting on the Internet: we are in particular interested in assessing the correlation, if any, between password strength, user activity levels, and the application domain.

ACKNOWLEDGMENTS

The authors are very grateful to Sebastian Porst and Roger Grimes who shared the set of MySpace passwords, to Cynthia Kuo, Sasha Romanosky, and Lorrie F. Cranor who shared their mnemonics dictionary, and to Matt Weir for his cooperation and valuable feedback.

REFERENCES

[1] R. E. Smith, *The Strong Password Dilemma*. Addison-Wesley, 2002, ch. 6.

[2] A. Adams and M. A. Sasse, "Users are not the enemy," *Commun. ACM*, vol. 42, no. 12, pp. 40–46, December 1999.

[3] S. Riley, "Password security: What users know and what they actually do," *Usability News*, vol. 8, no. 1, February 2006.

[4] R. Morris and K. Thompson, "Password security: a case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, November 1979.

[5] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proc. USENIX UNIX Security Workshop*, 1990.

[6] E. H. Spafford, "Observing reusable password choices," in *In Proceedings of the 3rd Security Symposium. Usenix*, 1992, pp. 299–312.

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

[8] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *CCS 2005*, 2005, pp. 364–372.

[9] J. A. Cazier and D. B. Medlin, "Password security: An empirical investigation into e-commerce passwords and their crack times," *Information Security Journal: A Global Perspective*, vol. 15, no. 6, pp. 45–55, 2006.

[10] S. Marechal, "Advances in password cracking," *Journal in Computer Virology*, vol. 4, no. 1, pp. 73–81, February 2008.

[11] D. Florencio and C. Herley, "A large-scale study of web password habits," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 657–666.

[12] "John the ripper - cracking modes," Retrieved on 29/07/2009. [Online]. Available: http://www.openwall.com/john/doc/MODES.shtml

[13] L. von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Commun. ACM*, vol. 47, no. 2, pp. 56–60, 2004.

[14] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in *Proc. CCS '02*, 2002, pp. 161–170.

[15] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Advances in Cryptology - CRYPTO 2003*, 2003, pp. 617–630.

[16] T. Wu, "A real-world analysis of Kerberos password security," in *Proc. NDSS 1999*, February 1999.

[17] M. Bishop, "Improving system security via proactive password checking," *Computers & Security*, vol. 14, no. 3, pp. 233–249, 1995.

[18] J. J. Yan, "A note on proactive password checking," in *Proc. NSPW '01*. ACM, 2001, pp. 127–135.

[19] R. A. Grimes, "MySpace password exploit: Crunching the numbers (and letters)," InfoWorld online article, November 2006. [Online]. Available: http://www.infoworld.com/article/06/11/17/47OPsecadvise_1.html

[20] S. Porst, "A brief analysis of 40,000 leaked MySpace passwords," Blog post, November 2007. [Online]. Available: http://www.the-interweb.com/serendipity/index.php?/archives/94-A-brief-analysis-of-40,000-leaked-MySpace-passwords.html

[21] C. Kuo, S. Romanosky, and L. F. Cranor, "Human selection of mnemonic phrase-based passwords," in *Proc. SOUPS '06*, 2006, pp. 67–78.

[22] B. Schneier, "Schneier on security: Choosing secure passwords," January 2007. [Online]. Available: http://www.schneier.com/blog/archives/2007/01/choosing_secure.html