Breaking our password hash habit

Why the sharing of users' password choices for defensive analysis is an underprovisioned social good, and what we can do to encourage it.

Cormac Herley
Microsoft Research
cormac@microsoft.com

Stuart Schechter
Microsoft Research
stus@microsoft.com

Abstract

Attackers only get better at guessing the passwords users will create under a given set of password-composition constraints. They learn more about users' password-selection behaviors each time they compromise a passsword, regardless of whether they obtain the password by breaching a password database, installing a key logger, phishing, or by guessing.

Defensive analysis of user-chosen passwords could similarly identify predictable password-selection behaviors and help us to prevent users from choosing predictable passwords. Alas, attempts to perform such analysis have been stunted by requirements to encrypt passwords irreversibly and by the indignation shown for those who would try to analyze the passwords their users choose.

We argue that encrypting passwords irreversibly has done more harm than good, providing a minimal short-term reduction in risk as opposed to reversible encryption, but imposing a severe cost on our ability to improve password defenses for the long term. As encrypting passwords is of little value if users continue to choose passwords that are easily guessed, our collective choice to blind ourselves to our users' passwords has made us collectively less secure. We argue that passwords should be encrypted so as to allow for offline defensive analysis.

While we believe there's a strong case to be made that the *social* benefits of defensive password analysis outweigh the risks, the *individual* cost/benefit tradeoff discourages users and firms from contributing passwords for defensive analysis. When choosing a password, an individual may benefit from analyses based on others' prior contributions, but does not benefit from contributing the password she chooses. However, she bears the risk should the password she has chosen is compromised as a result of contributing. This makes making free-riding on others' password contributions the security-optimizing strategy for her as an individual. To solve this free rider problem, we propose that systems that help to prevent users from choosing weak passwords (informed by prior users' passwords choices), require that those using the system contribute their newly-chosen passwords in return.

1 Introduction

Passwords are the dominant form of authentication for online accounts and services, and our reliance on passwords has grown as we use online accounts to work, communicate, interact with financial services, and even conduct our social lives. While the "death of passwords" has been predicted many times, a more realistic view suggests that the barriers to their replacement are profound and they are likely to be with us in great numbers for the foreseeable future [7]. The scale of our reliance on passwords makes it essential to understand and mitigate their vulnerabilities.

All forms of authentication have vulnerabilities, but the ubiquity of passwords has ensured that the public is aware of many of their shortcomings: attackers may observe a user entering a password either visually or electronically (*e.g.*, via a keylogger), trick the user into revealing the password (*e.g.*, by 'phishing'), capture the password in transit from the user to the server, or capture information stored about the password on the server. Alternatively, attackers may simply guess passwords until identifying the correct one.

Guessing attacks exploit the existence of common password-selection behaviors that result in common passwords. Some users choose uncommon passwords that are hard to guess, rendering their passwords prohibitively expensive to guess even if individually targeted by an attacker. However, a significant fraction of users choose passwords using predictable words or strategies that result in passwords that are themselves too easily predicted by attackers. An attacker seeking any single account within an organization to leverage for further attacks (a *beachhead*), or who sees value in obtaining and exploiting a subset of a services' accounts, can target all accounts at once in order to compromise the subset with *weak* passwords. For example, an attacker may try to login to all accounts using a statistical guessing attack: first using the password suspected to be most likely, followed by the next-most-likely password, and so on. For the purposes of defending against a beachhead attack, an organization is only as safe as its weakest password.

Some common password-selection behaviors are already well-known. One is to use a variant on the word `password` or phrases that connote a request for access (*e.g.*, `letmein` and `opensesame`). Advice given to users on how to construct passwords can have the unintended effect of stimulating common user behaviors, as can imposing requirements on how users choose passwords. For example, rules requiring the use digits and symbols and advice on using substitutions such as `@` in place of `a` may result in the common use of `P@ssw0rd`.

Unfortunately, attackers' knowledge of users' more predictable password-selection behaviors may be growing faster than ours, and far faster than our ability to discourage or prevent the selection of easily guessable passwords. Attackers can learn from users' password-selection habits each time they succeed in obtaining passwords through key logging, phishing, and by compromising password databases. While some attackers have publicly shared password databases from online services with minimal password-composition requirements (*e.g.*, RockYou), attackers who infiltrate corporations using stricter requirements have been less generous about sharing their findings. For example, we have many datasets from organizations that had relatively loose password policies (*e.g.*, six characters unrestricted) but no large sets from ones with restrictive policies (*e.g.*, eight characters, including upper, lower-case and special characters). It would be dangerous to assume that attackers share our ignorance regarding the common passwords chosen under the stricter password-generation requirements commonly in use.

So long as attackers may have knowledge used to predict users' password choices that we do not, there can be no such thing as a strong user-chosen password; there are only passwords that have no *known* weaknesses. A password may appear strong to us if it is derived from a concept, idea, or via a mechanism that is unknown to us, yet it may be predictable to an attacker who does not share our ignorance.

The analysis required to identify and understand common password-creation behaviors requires plaintext passwords, and faces two hurdles as a result. First, system designers are encouraged to store passwords using irreversible encryption, so that plaintext recovery can only be achieved for those passwords that can already be guessed. This prevents the identification of passwords that might be easy to guess once the behavior that caused them had been recognized, but that could not be guessed with existing knowledge. The use of encryption to prevent offensive analysis comes at the cost of also preventing defensive analysis. Second, while password sharing for defensive analysis is a social good, individuals and organizations obtain the greatest security by free riding on the analysis of others' passwords without contributing their own. Our goal is to tackle this free-rider problem so as to raise the overall security of all users of passwords.

Roadmap

This paper present an argument leading to a simple conclusion: data to study common password habits is a social good that poorly-telling users-aligned incentives prevent us from reaching. We begin, in Section 2, by presenting background on research in user-selected passwords and the adoption of password-storage practices that have hindered this research. We also explain why password-construction requirements fail to prevent weak passwords. In Section 3, we discuss three misconceptions about password-guessing attacks that have been hindered our progress in developing defenses against them. In Section 4, we explain why weak passwords cannot be prevented by suggestions or composition rules. Rather weak-password prevention systems must detect predictable behaviors and alert users to the presence of the weak behavior so that it

does not lead to the creation of a weak password. In Section 5, we explain defensive analysis requires a large number of passwords. In Section 6, we detail the free-rider problem faced by systems designed to prevent weak passwords in the long term, and offer a proposal to solve this dilemma in Section 7.

2 Password research: a bursty, checkered history

The practice of encrypting passwords with a user-specific irreversible function dates back at least to 1974 [4], and quickly became a recommended community-standard for the storage of passwords. Encrypting passwords was an important innovation because password databases need to be stored *online* so that they can be accessed to verify users' passwords during authentication. This availability requirement exposes password databases to compromise. Whereas the compromise of a plaintext password file results in the exposure of all users' passwords, the compromise of irreversibly-encrypted passwords does not. Attackers can guess passwords by computing the irreversible function on each guess, but this incurs a computational cost. When users choose passwords that are hard to guess, the computation required to crack them may delay the revelation of the password or even make cracking prohibitively expensive. Alas, if a user chooses a password that attackers are likely to guess fairly quickly, encryption provides little value.

The use of an irreversible (*a.k.a.* one-way) encryption function was preferred to symmetric encryption, as the symmetric key would need to be stored online; attackers who breached the system could recover the key along with the password database, decrypt the password database, and obtain the plaintext passwords. While the arrival of public-key cryptography in 1978 (starting with Rivest, Shamir, and Adleman's scheme [13]) made it possible to encrypt passwords with a public key kept online, but only decrypt them for analysis with a private key stored safely offline, this option does not appear to have been adopted.

In 1979, Morris and Thompson illustrated the incredible insight that can be obtained by analyzing such password data sets, showing that the great majority of passwords available to them were either dangerously short or consisted only of a word found in the dictionary [11]. To prevent users from future weak-password choices, Morris and Thompson created what may have been the first password-composition rules.

> The password entry program was modified so as to urge the user to use more obscure passwords. If the user enters an alphabetic password (all upper-case or all lower-case) shorter than six characters, or a password from a larger character set shorter than five characters, then the program asks him to enter a longer password. This further reduces the efficacy of key search.

2.1 The Dark Ages

While Morris and Thompson concluded that their password composition rules "make it exceedingly difficult to find any individual password" [11], there has been little evidence to support the claim. Similarly, a NIST standard [2], motivated by concerns that users "left to choose their own passwords will choose passwords that are easily guessed," proposes password-composition guidance and heuristics for estimating the strength of a password created under a given set of requirements. Neither the standard, nor subsequent research, has provided data to support these requirements or strength estimates.

In the decades that followed Morris and Thompson's work, there was relatively little detailed research into user password habits. Rather, academic research and industry have generated a bounty of numerous alternatives and proposals to replace password, none of which have made a dent in the ubiquity of password authentication [7]. As the number of computer users has grown by orders of magnitude, and the number of accounts requiring a password for each user has grown as well, there has been little growth in our knowledge of how users choose passwords or how to prevent them from making predictable choices.

The small collection of academic literature on password choice in the coming decades relied on small data sets which allowed only the most limited analyses. This has been the result of economic limitations and the availability of data. In 1999, Yan *et al.* [8] performed an experiment on first-year college students

which limited them to roughly one hundred participants. In studying a similar knowledge-based authenticator (secret questions) [14], we recruited over one hundred participants at a cost of over a hundred dollars per participant. Even studies that can be fully automated and use participants recruited through crowd-sourcing platforms face practical constraints that limit the potential sample sizes to the thousands (for example, see the recent contributions out of Carnegie Mellon University from Kelley *et al.* [9], Komanduri *et al.* [10], Shay *et al.* [17], and Ur *et al.* [18].)

2.2 The Renaissance

A recent renaissance in password research, spurred by releases of compromised passwords, gives further reasons to be suspicious of the password-composition rules and strength heuristics on which we've relied since the 1970s.

This renaissance began in 2006, when a collection of 34,000 myspace passwords (purportedly the result of phishing attacks) was leaked and widely distributed to the point of becoming effectively public. This appears to have been the first widely-publicized leak of a password collection large enough to permit meaningful statistical analysis. These analyses revealed that some popular choices, such as `password` and `abcdef`, were indeed very common [16]. Some of the common behaviors revealed appeared to have developed in response to composition rules, such as adding a trailing digit or special character to the end of a password (*e.g.*, `password1` or `password!` instead of `password`). The analysis also revealed common password-selection behaviors that would likely have gone undetected without data. For example, `blink182`, the name of a popular band which happens to satisfy the requirement of containing digits, was one of the most popular myspace passwords. There were several medium-sized (*i.e.*, tens of thousands or so) leaks of password sets in the next few years.

In 2007, Florêncio and Herley [5] examined password habits of half a million toolbar users. While they documented several interesting features such as length, character composition, and frequency of re-use they did not have access to plaintext; thus, they did not study how common passwords are, and what strategies people use to create them.

Zhang *et al.* [22] tried guessing new passwords following password-resets by using variants of the previous passwords. They succeeded 41% of the time in an offline attack, and 17% of the time using an online attack. Clearly, in many cases, new passwords are created by making minor modifications to the old ones. This reveals that a policy (*i.e.*, mandatory password changes at fixed intervals) that has been in widespread use for many years is falling far short of achieving its design goals.

The renaissance's momentum was provided a rapid, if unintended, acceleration in December 2009, when hackers breached the servers of RockYou and published the database of plaintext passwords for the site's 32 million users. This corpus of user-chosen passwords was three orders of magnitude larger than that leaked from myspace. Weir *et al.* [21] were among the first to publish academic analyses of the data set and its implications for password security. They found that the NIST heuristics for approximating entropy [2] grossly overestimate the difficulty of guessing the weakest passwords created with password-composition rules.

For example, the most common password in the RockYou dataset, `123456` accounts for about 0.9% of all accounts. This is very weak, indicating that an attacker gets almost one account in a hundred with one guess each. Yet, among the subset of RockYou passwords that are at least 8 characters long and have upper, lower-case, special characters and digits the most common password, `P@ssw0rd`, accounts for 0.88% of accounts. Thus, it's arguable that forcing users to comply with onerous composition policies does not improve the resistance to a beachhead attack.

Two years later, Bonneau published an analysis of a corpus of 70 million Yahoo!-account passwords, more than twice the size of the RockYou data set [1]. Unlike prior analyses as this scale, this work was performed with the consent of the site from which the passwords originated. Bonneau was permitted to install code that briefly analyzed a password when it was available as plaintext to servers during login, but did not store the plaintext password for later study. Rather, the code recorded characteristics about the password (*e.g.*, length and the types of characters present), a one-way hash of the password that was

consistent across users so as to determine if two users had chosen the identical password, and demographic information about the user. Thus, Bonneau was able to study patterns of frequency, and proposed a measure of strength that correlates better with attacker success than any of the other known measures. Even though the one-way encryption did not identify actual passwords, the histograms of hashes would give a prospective attacker significant insight into the likely success of online guessing attacks. The willingness of Yahoo! to allow such a study represents an important shift in prioritizing long-term defensive understanding over short-term efforts to minimize user risk.

3 Misconceptions muddy thinking about password-guessing defenses

Discussions of defense against password-guessing are plagued by three common misconceptions: (1) that the problem can be modeled as one of making passwords 'stronger'; (2) that the best way to defend against attackers who would compromise the password database, is to store passwords via a one-way function and enforce the selection of passwords 'strong' enough to resist offline-guessing attacks on that function; and (3) that we can rely on data released from breaches to teach us how to detect and prevent weak passwords.

3.1 Password security is measured by 'strength'

The problem of preventing password-guessing attacks is often reduced to discussions of "increasing password strength," where 'strength' implies resilience to guessing. This conceptualization may be responsible for many ill-conceived metrics of users' collective vulnerability to guessing attacks and defenses against guessing attacks.

One resulting misconception is that password strength is measurable and that a user-chosen password can be measurably strong. In fact, a password that appears strong to one observer may be predictable to an attacker who is more familiar with the mindset or dialect of the user who chose it. We can at best measure and detect whether passwords exhibit known weaknesses.

Another 'strength'-related misconception is that increasing the average password strength (decreasing the average prevalence of weakness) has a meaningful impact on security. Consider an organization with ten user accounts. If the organization were to require the user who already had the best password to add 100 random characters to it, an estimate of mean strength would show that the organization had just become orders-of-magnitude more safe. In fact, the organization would likely become much more safe by adding a single random character to the weakest password.

Metrics of average password strength or weakness also confuse analysis of attacks that target a single high-value user, or some subset of these users. Users with low-value accounts may skew the average by putting correspondingly lower investments into choosing good passwords.

The most restrictive password policies are often imposed on employees of companies, governments, and other organizations by their IT departments. These organizations have much to fear as the compromise of a single employee's password can provide attackers with a *beachhead* into the organization from which to launch further attacks. Such an attack requires the compromise of only a single employee account, and so defending against it is a weakest link problem [19]. The only way to increase security against such an attack is to replace the weakest password with one that is not known to be unacceptably weak, and continue doing so until no password is known to be unacceptably weak. A very successful security strategy need not necessarily result in any change in median 'strength' (or weakness) and may have a very small change on mean 'strength' (or weakness).

Not all users will have to change their password to protect against a beachhead attack. In fact, there's reason to believe most users already choose passwords that do not exhibit dangerous levels of weaknesses. Of the $32,603,388$ user accounts whose RockYou passwords were compromised, 36.45% ($11,884,632$ users) of accounts had a unique password and 65.89% ($21,478,750$ users) had a password that occurred with frequency less than one in a million (32 occurrences or fewer). One could presumably expect users to choose even better passwords on sites they believed to be security-critical or that required more than

six characters. One common misconception about 'strengthening' passwords is that any effort to enforce stronger passwords will result in a proportional increase in users forgetting their passwords. If those who are already choosing passwords that aren't classified as dangerously weak do not need to change what they are doing, the usability cost of removing weak passwords need not be so great.

3.2 The best defense against password-file breaches is a strong hashed password

Even if the weakest passwords were orders of magnitude harder to guess than they are today, encrypting with a one-way function and user-specific salt could not prevent an attacker with access to the password database from guessing some fraction of these passwords.

Consider if were were to take radical password-strength and hashing measures to try to protect ourselves in the event of a password database breach. We might choose a hash function that incurs a tenth of a second of latency on a dedicated processor core. While this would increase login latency in a manner that is at the edge of being perceivable by users, and expose authentication services to greater risk of denial-of-service attacks, it would at best limit attackers to issuing ten a guesses per core per second. Let's further assume that, through great strides in weak-password prevention, we are able to ensure that the weakest passwords require a billion guesses to crack. This is equivalent to enforcing a unique password requirement on a billion users. Even with these measures in place, an attacker with a million machine botnet (with an average of our four cores per machine) could issue over 2 billion guesses per minute and expect to crack a password every twenty-five seconds.

Alas, the above analysis is actually an optimistic one, as few defenders go to such extreme measures to protect password databases. While it is possible to use increase the computation required by hash functions over time to keep up with advances in hardware (see Provos and Mazières [12]), many fail to make the effort to do so. Those services that still rely on Windows NTLM-hashed passwords (against Microsoft's recommendations [3]) could see *all* eight-character passwords cracked within six hours by a cluster of only five GPU-enhanced servers [6]. While the password database from LinkedIn was hashed (though not salted), hackers cracked more than 60% of passwords [20].

The importance the security community places on hashing, paired with the limit protection it offers against password-database compromise, might lead one to believe that the only explanation for the prevalence of this solution is the lack of better alternatives. In fact, there are better alternatives.

There already exist *hardware security modules* (HSMs) that perform cryptographic operations on keys stored within the hardware and that do not provide functionality to read (extract) these keys.[1] These HSMs allow keys to be written, but not read, and are already used for such purposes as key management (storing master keys that sign other keys) and SSL acceleration. Using this hardware, the hash function could be paired with a firm-specific private salt (a key) that cannot be extracted via a compromise of the system in which the hardware is installed. The resulting hash function would not only be one-way, but could only be calculated with the hardware. For distributed authentication systems, multiple copies of the hardware have the same firm-specific salt inserted into them before they are put online. Since the hardware does not provide functionality for reading (extracting) the firm-specific salt, an attacker who compromises the password file will only be able to crack passwords by issuing guesses remotely to the hardware at whatever rate the hardware permits. Once the compromise is discovered and the attacker loses access to the hardware, he can no longer issues guesses.

The same hardware technique can be used to protect passwords stored using public-key encryption to allow for defensive analysis. Before encrypting the plaintext password with the public key, the hardware appends to it the hash of the password calculated using the firm-specific private salt.

Unlike attempts to strengthen passwords, hardware security modules can be deployed without imposing a burden on end users. However, they cannot protect against online guessing attacks. Thus, the goal

[1] See, for example: `http://www.thales-esecurity.com/products-and-services/products-and-services/hardware-security-modules` and `http://www.safenet-inc.com/data-protection/hardware-security-modules/ssl-accelerators/`

of weak-password prevention should be to protect against *online* guessing attacks, whereas better (less burdensome) options are available to protect against *offline* guessing attacks.

3.3 We can rely on the kindness of thieves to help us prevent weak passwords

The recent flow of breaches has caused the public release of password databases from tianya, stratfor, Gawker, IEEE, LinkedIn and RenRen. Ironically, the thieves who have compromised large databases of user passwords may have produced a greater net social benefit than law-abiding academic research teams whose work was constrained by the need to protect users. Research teams such as Florêncio and Herley [5], Zhang *et al.* [22], and Bonneau [1] could examine only those hypotheses that could be anticipated in advance and that could be tested with minimal risk. They also could not publish the data used in their analyses. In contrast, the criminals who breached RockYou and other sites provided the research community with public data sets with which researchers could test a wide range of previously-unanticipated hypotheses and replicate others' results.

Prior to the release of passwords from myspace, RockYou, and the stream of breaches since, defenders' self-imposed security restrictions ensured they had less insight into users' passwords choices than many attackers likely had. Attackers who obtained plaintext password databases may have long known what the research community discovered only recently. Other attackers may have come to the same knowledge by harvesting passwords through phishing, keylogging, by trying to login to online accounts with guessed passwords (*a.k.a.* online dictionary attacks), or guessing candidate passwords against compromised encrypted password databases (*a.k.a.* offline dictionary attacks).

Asymmetric information about users' password-selection behaviors allows attackers to guess the most common passwords that we don't yet know how to prevent. It would be foolish to assume that the thieves who steal passwords will continue to share their bounty with the research community and give up the advantage that comes from asymmetric information. It would also be foolish to assume that the information we've learned from past disclosures now puts the research community on equal footing with attackers. Publicly-disclosed password lists have come primarily from websites that users may not have considered worthy of a strong password and that have relatively relaxed password-composition rules. RockYou, for example, accepted six character passwords such as `abcdef` and `123456`. Some users will re-use passwords from sites with password-composition rules at these laissez-faire sites, but they are a minority and those that engage in this practice may not provide a representative sample.

There has not, as yet, been a large published dataset from a site that requires more stringent password-composition rules, or that forces periodic password changes. Thus, we know that users respond to password-composition policies using such predictable passwords as `P@$$w0rd`, but we do not know which choices are the most common or just how common they are. Furthermore, we do not know the impact of other password constraints placed on users. Since each constraint is likely to stimulate common strategies among users who required to satisfy it, attackers who compromise passwords generated under these constraints will know about weak passwords that defenders without access to similar data cannot know about.

4 Detecting weak passwords is the only way to prevent them

The dominant means of preventing weak passwords has been to employ proactive interventions that prescribe desired behaviors. For example, rules or guidance specify in advance that passwords should be at least a certain length, use different character sets (uppercase, lowercase, digits, and punctuation), and even dictate that using certain types of characters appear in certain positions.

4.1 Why proactive interventions fail

These *proactive prescriptive* interventions may prevent users from choosing the subset of common passwords that fail to meet the prescribed requirements, but they may also stimulate common behaviors as users react to them. Some common passwords may be prevented by requiring the presence of digits (*e.g.*, `changeme`)

and a password length of at least eight characters (*e.g.*, `123456`), but others may increase in prevalence as users find common ways to satisfy these requirements (*e.g.*, `changeme1` and `12345678`). Similarly, requiring special characters may eliminate other common passwords, but cause users to respond by looking at their available character choices cluster around concepts they associate them. For example, users who fixate on the '@' symbol may cluster around words or phrases that contain the sound 'at' (*e.g.*, `thec@&theh@`); the dollar symbol may suggest concepts of money (*e.g.*, `otherpeople's$`); and the exclamation point may only increase the already-common propensity to employ profanities in passwords.

Even when not required, well-intentioned guidance that suggests techniques for creating passwords can unintentionally suggest common behaviors that weaken passwords instead of strengthening them. For example, the suggestion to substitute similar-looking characters into words (*e.g.*, use '$' in place of 's') may inspire a subset of users who think they are clever to choose a password that both uses substitution and contains the word substitution: `$u8$717u710n`. Such a password is an unintended consequence of guidance, and would not likely be popular without such guidance.

Proactive proscriptive interventions, which tell users what behaviors to avoid *before* they choose a password, are both impractical and potentially counterproductive. Even with our limited knowledge of user behavior today, we cannot expect users to take the time to read and understand the sample set of dangerously-common behaviors we already know about. Even the most common behaviors are popular with a small fraction of users (*e.g.*, roughly 1% would use `password`) and so, for any given user, almost every proscription would discourage a behaviors she had not previously considered engaging in. It's even possible that more users will be inspired to try a variant of a proscribed behavior than would have considered doing so if they were left unaware that such a behavior existed. For example, a proactive prescription on `password` might inspire some fraction of users who had previously had no plans to use a variant on the word to choose `drowssap` (`password` spelled backwards).

4.2 Reactive proscriptive intervention

Reactive proscriptive intervention is presented only upon detecting predictable (and therefore weak) behavior. The intervention is proscriptive in that its goal is to cause users to abort this behavior. One well-known reactive proscriptive intervention is to search for dictionary words within a password and forbid their use.

A proscriptive approach need not forbid all common behaviors outright. Rather, the intervention need only ensure that predictable behaviors do not result in predictable passwords. If a password contains a substring that is hard to guess, it will not be harmed by the addition of more password material even if those characters are easy to guess. For example, a password may not be made much stronger by appending the predictable string `password` to it, but it will not be made weaker either.

Furthermore, behaviors that are unacceptably predictable on their own may be acceptable when used multiple times or in composition with each other. Choosing a dictionary word at random may not result in a password that is sufficiently hard to guess, but choosing three dictionary words at random may result in a password that is quite hard to guess.

We previously proposed the reactive intervention of checking to see if a newly-chosen password is already common among existing users, and proscribing the use of those passwords that proved predictably popular [15]. This simple approach relies on safety in numbers, and to ensure that the number of expected guesses to obtain access to the account with the weakest password is n, the system must have at least $2n$ accounts. This approach fails to prevent predictable user-specific behaviors that may not always result in common passwords, such as using a variant of one's username as a password.

A popularity-based weak-password prevention mechanism may also fail to protect against attacks that exploit language common only to a local dialects. Dialects are not just geographical, but can be specific to subcultures and organizations. Organizational dialects may include acronyms, project code names, and other words that are predictably popular within the organization but would not be popular among the general population. Attackers who know a local dialect could exploit that knowledge to guess likely passwords among this small subset of users.

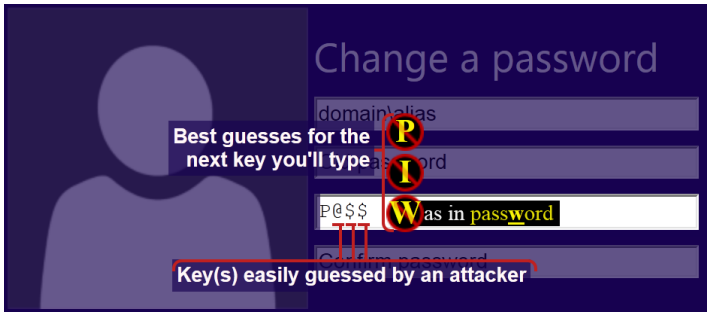

Figure 1: *Telepathwords* is a prototype of a reactive proscriptive system to deter users from choosing weak passwords. The system provides feedback to inform users if they are about to type a predictable character, or have already done so. Administrators can require passwords to have a minimum number of hard-to-guess characters.

When it is not possible to evaluate weakness purely on the popularity of a password in a large data set, defenders can instead try to identify and prevent predictable behaviors and substrings. An example of such a system is a prototype we have been developing in collaboration with Saranga Komanduri of CMU. The system is called *telepathwords*, as it illustrates the presence of suspected-weak behaviors by displaying predictions for the next character the user *intends* to type (but has not yet shared with the system). The system only counts those characters it could not predict when assessing a user's progress towards a minimum-length requirement. Thus, the *telepathwords* prototype does not prevent weak behaviors, but instead seeks to ensure the password contains a sufficient number of characters that do not appear to be the result of predictable behaviors.

While not measuring Shannon entropy, a metric commonly referred to in discussions of password strength, the approach uses the similar observation that the best way to determine the contribution of an additional character is to determine how likely it could be guessed if the characters before it were already known. The prototype makes predictions using a dictionary of passwords revealed in public compromises, a language model of common words and phrases (n-grams), and a list of common character transforms (*e.g.*, 'a' to '@'). It also recognize common behaviors such as interleaving two predictable strings (`p1a2s3s4w5o6r7d8`), moving a finger over adjacent characters (`asdfghjkl;`), and repeating characters (`********` and `fourfourfourfour`). Future versions could allow local dialects to be integrated into the set of common words and phrases. Figure 4.2 shows the prototype in action.

The limitation of reactive weak-password prevention systems like *telepathwords* is that they cannot detect or proscribe unanticipated weak behaviors that go undiscovered. Furthermore, we cannot discount the possibility that a system like telepathwords will stimulate common responses leading to new categories of predictable behaviors. For any reactive system to work, we must be able to continue to identify common behaviors that evolve after the system is deployed. (Our prior popularity-based weak-password prevention scheme [15] automatically recognizes *passwords* that may evolve from users exposure to it, but it cannot recognize the more general underlying *behaviors* that inspire users to choose them.)

5 Weak-password prevention requires analyzing many passwords

Given that some password-selection behaviors can only be recognized by examining the passwords that users choose in response, defensive analysis of plaintext passwords will be essential to the efficacy of any behavior-based weak-password prevention system.

5.1 Common behaviors revealed through identical passwords

The most straightforward way to detect common behaviors is to examine popular passwords, starting from the most common password and proceeding to successively less common passwords. Our ability to detect

any common behavior with this method is limited by the likely prevalence of passwords resulting from that behavior.

The selection of identical passwords for two user accounts does not necessarily mean their behavior is truly common among the larger population. The two accounts may actually belong to the same user or users who have deliberately shared a password. Other such 'collisions' may occur simply due to chance; even if users choose passwords uniformly and at random from a space of size n, the birthday paradox will cause chance collisions to occur after roughly $\sqrt{n}$ passwords have been observed.

To illustrate the difficulty of gathering sufficient data to identify common behaviors through common passwords, consider a password has probability p of being chosen by any individual user. The probability that a corpus with N users will see it twice or more is $1.0 - \text{binocdf}(1, N, p)$. To be concrete, if a password is chosen with probability p it will appear twice or more with probability 26% in a corpus of $N = 1/p$ and 99.9% in a corpus of $N = 10/p$ users. Thus, patterns that are common enough to be very useful to an attacker (*e.g.*, $p = 10^{-4}$, indicating one in ten thousand) may not be common enough to be visible to the defenders. This is especially true if the attacker observes the success rate of a password across many different sites, while the defender observes it across one (if he examines the histogram of user passwords) or none (if he stores the passwords in encrypted form).

To achieve the promise of passwords resilient to a million guesses, as NIST guidance [2] would have lead us to believe possible using password-composition rules, we will likely need to analyze tens of millions of passwords.

5.2 Common behaviors revealed by common password substrings

Fortunately, plaintext analysis also allows us to identify common passwords by looking for common substrings. Additional commonalities can be identified by performing canonicalization in advance of this search, such as by turning uppercase letters into lowercase and treating similar looking characters as if they were equivalent.

While it is harder to formally bound the effectiveness of such an analysis, large data sets are still required.

6 Building a password corpus is a free-rider problem

The question of whether to allow defensive analysis is one of costs and benefits. Let us call the benefits of having a corpus of n passwords available for defensive analysis $B(n)$.

Contributing a password is not without cost. There exists the small possibility that the decryption key used to analyze passwords (which should be stored offline and well guarded) could be compromised, that those charged with defensive analysis of the passwords might fail to protect them, or that the attackers will benefit from learning about common behaviors by examining the weak-password detection systems. We represent the cost of collecting and storing a corpus of n real passwords as a function $C(n)$.

The expected marginal benefit of each additional password is smaller than the last, as it is likely to reveal behaviors that are statistically less common. In other words, the second derivative of $B(n)$ is negative. In contrast, to a first approximation, each additional password contributed puts an additional account at risk, and so $C(n)$ grows roughly linearly.

The net total benefit of contributing n passwords can be written as:

$$T(n) = B(n) - C(n)$$

The net marginal benefit to the participants of contributing the nth password in the corpus is the first derivative of $T(n)$. Approximating $T^{'}(n)$ by a discrete function:

$$T^{'}(n) = [B(n) - B(n-1)] - [C(n) - C(n-1)]$$

The social welfare is increased by the contribution of passwords for defensive analysis until the size of the corpus reaches some threshold n_t such that the marginal net benefit of the contribution to society is zero: $T^{'}(n_t) = 0$.

We have argued that some password-selection behaviors are common enough to be worth discovering and preventing, but not so common as to be discoverable with only a small password corpus. In other words, if contributed passwords can be secured well-enough to keep the risk low, n_t is likely in the millions, tens of millions, or even higher.

Alas, regardless of the value of n_t that maximizes social benefit, contributions of passwords for collective analysis are likely to be underprovisioned because the choice to contribute a password for defensive analysis may not be attractive to the individual making the contribution. The problem is that while everyone benefits from a contribution, the individual contributor sees only a negligible fraction of that contribution's benefits but incurs its full cost.

In other words, society benefits when the security benefits of the contribution, which benefits everyone, outweigh the costs:

$$B(n) - B(n-1) > C(n) - C(n-1)$$

but the individual only comes out ahead when his tiny share of the security benefits outweigh his full share of the cost (security risk)

$$\frac{B(n) - B(n-1)}{n} > C(n) - C(n-1)$$

More generally, an organization that could contribute a passwords to bring the total size of the corpus to n only comes out ahead when:

$$\frac{a}{n}\left[B(n) - B(n-a)\right] > C(n) - C(n-a)$$

Alas, even if we are able to overcome the conventional wisdom that passwords can be sufficiently strengthened via composition rules, that passwords must be stored using an encryption algorithm that cannot be reversed, and that the benefits of allowing defensive analysis outweigh the costs, we cannot expect organizations and individuals to volunteer to contribute their passwords for analysis. As with voting in a large democracy, the act of contributing cannot be justified by one's chance of having an impact on the outcome, let alone benefiting personally.

7 Overcoming freeloading requires trust and the right incentives

The perceived risk of contributing a password for defensive analysis depends on how much the prospective contributor trusts those charged with protecting the passwords, performing the analysis, and integrating findings into weak-password prevention systems. The added risk of contributing may be smaller if the prospective contributor is already trusting the provider of the weak-password prevention with the secure operation of the larger authentication system or the operating system. The organization that obtains these passwords and performs defensive analysis can further reduce perceived cost by being transparent about the process through which it protects passwords in storage and during analysis (*e.g.*, separating passwords from the identity of the users who created them and the services they were created to authenticate to), about who it grants access to perform analysis, and about how it handles requests from governments that may want to access these data.

Classical economics would suggest that the positive externality of contributing a password could be internalized through a payment to the contributor, presumably proportional to (but smaller than) the expected social benefit. In practice, there are many reasons to suspect that this would not result in the behavior predicted by a model of utility-maximizing rational economic behavior. First, humans do not easily convert the currency of risk to dollars. Second, the officers charged with making security risk decisions (*e.g.*, CISOs) are unlikely to be rewarded for the incoming cash flows and might fear being punished for introducing a new security risk—even if that expected dollar cost of that risk is small relative to the

payment. Finally, firms could not expect investors or customers to accept the justification of economic rationality for taking a new security risk. Rather, a rational cost-benefit analysis is more likely when the cost (a potential increase of security risk) is more than offset by a benefit of the same currency—a larger decrease in security risk.

For both password-popularity and behavior-based weak-password prevention schemes, we propose that the best way to prevent free riding is to make the contribution of a user's new password the price of using the weak-password prevention system. This would change the organization's decision function for contributing a additional passwords to the corpus to (since $B(0) = 0$):

$$\frac{a}{n}B(n) > C(n) - C(n-a)$$

Assuming a linear cost function for which each additional password incurs a cost c:

$$\frac{a}{n}B(n) > ac \tag{1}$$

$$\frac{B(n)}{n} > c \tag{2}$$

$$\tag{3}$$

This approach is *fair* to each contributor in that each password contributed receives an equal proportional share of the social benefit and cost.

For firms that already trust their security to the provider of the weak-password prevention system – relying on it to provide safe code and services – the orders-of-magnitude reduction in the weakness of firms' weakest passwords will more than offset the added risk.

One way to further increase the risk/benefit trade-off, and further encourage firms to contribute passwords for analysis, would be to bundle the weak-password prevention system with a hardware security module (as discussed in Section 3.2) to reduce the risk of password database compromises. This additional benefit of participating of protecting a password with an HSM (B_{HSM}) so as to reduce the risk of offline dictionary attacks may further offset any risk of participating, changing the decision function to.

$$\frac{B(n)}{n} + B_{\text{HSM}} > c \tag{4}$$

$$\tag{5}$$

No matter what incentives are provided, some prospective users of the weak-password prevention system may still not have enough trust in the system to be willing to contribute their passwords. One way to accommodate such users is to allow them to opt-out of contributing by paying a fee at or above the social value of their forgone contribution. This fee could be used to fund the development of the system or distributed to the systems' other users to help address the reduction in fairness. Alas, in addition to any real or perceived reductions in fairness which may discourage others from contribution, allowing users to opt out of contribution reduces the system's potential efficacy. The firm that opts out will likely see the greatest reduction in efficacy, as the weak-password prevention system will be blind to any firm-specific behaviors or dialects that could only be detected through defensive password analysis. Similarities among the types of entities that might be most tempted to opt out (security firms, intelligence agencies, and so on) might amplify this effect.

Another approach to accommodating those wary of contributing passwords would be to instead collect newly-expired passwords when newly-created passwords replace them. This would introduce delays into weakness discovery, especially as it relates to terms and phrases that result from fast-moving trends. Users who are not required to change their passwords periodically might never contribute one. Finally, given the frequent correlation between expired and new passwords, collecting newly-expired passwords is not significantly less risky than collecting current passwords.

8 Conclusion

We have argued that overestimates of the benefits of password-composition policies and irreversible hashing, paired with a corresponding underestimates of the benefits of defensively analyzing plaintext passwords, have stunted improvements in password security. Rather, preventing weak passwords requires an approach that is both reactive and proscriptive, and relies on our ability to detect dangerously-common passwords and password-construction behaviors by analayzing a large representative corpus of plaintext passwords. While contributing passwords for defensive analysis by weak-password prevention systems is a social good, users are individually best served by using the weak-password prevention systems while free riding on analyses of passwords contributed by others. If we do not correct for this free-rider problem, our knowledge of users' password-selection habits will continue to trail that of attackers and become less accurate each time we iterate our systems in an attempt to eliminate the common behaviors we do know about. All users will suffer as weak-password prevention systems fail to keep up with attackers' growing knowledge of how users choose passwords. We have shown that requiring users to contribute their newly-chosen passwords in return for receiving assistance from a weak-password prevention system addresses the free-rider problem in manner that achieves an important form of fairness, in which as the proportion of the passwords a contributor protects (its share of the benefits) are are proportional to its contributions (its share of the passwords placed at risk).

References

[1] Joseph Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the 20012 IEEE Symposium on Security and Privacy*, Washington, DC, USA, May 20–23 2012. IEEE Computer Society.

[2] William E. Burr, Donna F. Dodson, and W. Timothy Polk. Electronic Authentication Guideline. In *NIST Special Publication 800-63*, 2006. `http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf`.

[3] Microsoft Corporation. `http://msdn.microsoft.com/en-us/library/cc236715.aspx`.

[4] Arthur Evans Jr., William Kantrowitz, and Edwin Weiss. A user authentication scheme not requiring secrecy in the computer. *Communications of the ACM*, 17(8):437–442, 1974.

[5] D. Florêncio and C. Herley. A Large-Scale Study of Web Password Habits. *WWW 2007, Banff.*

[6] Dan Goodin. 25-gpu cluster cracks every standard windows password in < 6 hours. *Ars Technica*, December 9 2012. `http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/`.

[7] C. Herley and P.C. van Oorschot. A Research Agenda Acknowledging the Persistence of Passwords. *IEEE Security & Privacy Magazine. Jan. 2012.*

[8] J. Yan and A. Blackwell and R. Anderson and A. Grant. Password Memorability and Security: Empirical Results. *IEEE Security & Privacy*, 2004.

[9] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio López. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. *IEEE Symposium on Security and Privacy*, 0:523–537, 2012.

[10] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect

of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2595–2604, New York, NY, USA, 2011. ACM.

[11] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[12] Niels Provos and David Mazières. A future-adaptive password scheme. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, USENIX ATC 1999, pages 32–32, Berkeley, CA, USA, 1999. USENIX Association.

[13] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[14] Stuart Schechter, A. J. Bernheim Brush, and Serge Egelman. It's no secret: Measuring the security and reliability of authentication via 'secret' questions. In *Proceedings of the 2009 IEEE Symposium on Security and Privacy*, Washington, DC, USA, May 17–20 2009. IEEE Computer Society.

[15] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *The 5th USENIX Workshop on Hot Topics in Security (HotSec)*, August 10 2010.

[16] Bruce Schneier. Schneier on security: Real-world passwords. December 14 2006. `http://www.schneier.com/blog/archives/2006/12/realworld_passw.html`.

[17] Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Blase Ur, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Correct horse battery staple: exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 7:1–7:20, New York, NY, USA, 2012. ACM.

[18] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L. Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. How does your password measure up? the effect of strength meters on password creation. In *Proceedings of the 21st USENIX Security Symposium*, August 8–10 2012.

[19] Hal R. Varian. Sytem reliability and free riding. In *The First Workshop on the Economics of Information Security*, May 2002.

[20] Jaikumar Vijayan. Hackers crack more than 60 *ComputerWorld*, June 7 2012. `http://www.computerworld.com/s/article/9227869/Hackers_crack_more_than_60_of_breached_LinkedIn_passwords`.

[21] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 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*, CCS '10, pages 162–175, New York, NY, USA, 2010. ACM.

[22] Y. Zhang, F. Monrose and M. K. Reiter. The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proc. CCS*, 2010.