Pushing on string: the "don't care" region of password strength*

Dinei Florêncio Cormac Herley
Microsoft Research, Redmond, USA

Paul C. van Oorschot
Carleton University, Ottawa, Canada

Abstract. We examine the efficacy of tactics for defending password-protected networks from guessing attacks, taking the viewpoint of an enterprise administrator whose objective is to protect a population of passwords. Simple analysis allows insights on the limits of common approaches, and reveals that some approaches spend effort in "don't care" regions where added password strength makes no difference. This happens either when passwords do more than enough to resist online attacks while falling short of what's needed against offline attacks, or when so many accounts have fallen that an attacker gains little from additional compromises. Our review of tools available to improve attack-resistance finds, for example, that compelling returns are offered by password blacklists, throttling and hash iteration, while current password composition policies fail to provide demonstrable improvement in outcomes against offline guessing attacks.

1 Introduction

Suppose a system administrator is tasked with defending a corporate, government or university network or site. The user population's passwords are targeted by attackers seeking access to network resources. Passwords can be attacked by guessing attacks—online or offline—and by capture attacks, *i.e.,* by non-guessing attacks. How to choose among password-attack mitigations is a practical question faced by millions of administrators. Little actionable guidance exists on how to do so in a principled fashion. Sensible policies must consider how to protect the *population* of user accounts. What is the best measure of the strength of a population of passwords? Is a good proxy for the overall ability of the network to resist guessing attacks the average, median, strongest or weakest password? Slogans such as suggesting that all passwords should be "as strong as possible" are too vague to guide action—and also suggest that infinite user effort is both available and achievable.

2 The compromise saturation point

Network compromise thought experiment. To begin, consider an administrator who observes unusual behavior on his network and suspects that some accounts have been compromised. If he thinks it's just a few accounts, and can identify them, he might just block access to those. However, if he can't be sure that an account is compromised until he sees suspicious activity, it is hard to figure out the magnitude of the problem. Should he block access to all accounts and trigger a system-wide reset? If only a few accounts have been compromised, perhaps not; if half of them have, he almost certainly should. What about a 1% compromise rate, or 5% or 10%? At what point is global reset the right answer?

Suppose our hypothetical administrator resets all accounts. There still remains the question: How were the credentials obtained in the first place? If the door that led to the compromise remains open (e.g., undetected keyloggers on several machines) then nothing improves after a system-wide credential reset. On the other hand, if the credentials were compromised by guessing, then a reset (at least temporarily) helps, and a change in the policies that allowed vulnerable passwords might be in order. But even if he concludes that password guessing was the attack channel, was it online guessing? Or, somehow, did the attacker get hold of the password hash file and succeed with an offline guessing attack?

When attacks on passwords succeed, the specific attack channel is not necessarily clear—it is not obvious whether the compromised accounts had weak passwords, were spear-phished or were drive-by download victims. Further, if it is not known which accounts have fallen, it may be best to reset all of them—even those not compromised. To facilitate more precise reasoning, let α

*Aug.24, 2016. This is the author's copy for personal use. A version of this paper will appear in *C.ACM*, Nov.2016.

be the fraction of credentials under attacker control—whether or not yet exploited. Thus, when $\alpha = 0.5$, half of the accounts (passwords) have already fallen to an attacker. At that point, would the administrator consider the network only 50% compromised, or fully overrun? This depends of course on the nature of the network in question. If a compromised account never has implications for any other account, then we might say that the damage grows more-or-less linearly with α. However, for an enterprise network a compromised account almost certainly has snowballing effects [3]: a single credential might give access to many network resources, so that the damage to the network grows faster than α (a possible curve is shown in Fig.1). At $\alpha = 0.5$, a system is arguably completely over-run. For many enterprise environments in this scenario, there would be few if any resources that the attacker can't access; in many social networks, the network value would approach zero, as spam would probably render things unusable; access to (probably well under) 50% of email inboxes likely yields a view to almost all company email, as an attacker requires access to only one of the sender or recipient(s) of each message.

All password-based systems must tolerate *some* compromise; passwords will be keylogged, cross-site scripted and spear-phished, and a network unable to handle this will not be able to function in a modern threat environment. As an attacker gains more and more credentials in an enterprise network, she naturally reaches some *saturation point* after which the impact of additional fallen credentials is negligible, having relatively little effect on the attacker's ability to inflict harm. The first credential gives initial access to the network, the second, third and fourth solidify the beachhead, but the benefit brought by each additional credential decreases steadily. The gain is substantial when k is small, less when large: the second password guessed helps a lot more than the hundred-and-second.

Let α_{sat} be the threshold value at which the attacker effectively has control of the password-protected part of the system, in the sense that there is negligible marginal gain from compromising additional credentials. That is, if an attacker had control of a fraction α_{sat} of account credentials, there are very few resources that she could not access; so the difference between controlling α_{sat} and $(\alpha_{\text{sat}} + \varepsilon)$ is negligible. In what follows, our main focus is enterprise networks, to consider possible values for α_{sat}.

There are a variety of tools that attackers can use, once they have one set of credentials, to get others. Phishing emails that originate from an internal account are far more likely to deceive coworkers. Depending on the attacker's objective, a toehold in the network may be all that she requires. The 2011 attack on RSA [1] (which forced a recall of all SecurID tokens) began with phishing emails to [13] "two small groups of employees" none of whom were "particularly high profile or high value targets." Dunagan et al. [3], in examining a corporate network of over 100k machines, found that 98.1% of machines allowed an outward snowballing effect allowing compromise of 1,000 additional machines. Edward Snowden was able to compromise an enormous fraction of secrets on the NSA network starting from just one account [14]. Given that RSA and the NSA (organizations that we might expect to have above average security standards) experienced catastrophic failures caused by handfuls of credentials in attacker hands, we suggest a reasonable upper bound on the saturation point for a corporate or government network is $\alpha_{\text{sat}} \approx 0.1$; saturation likely occurs at much lower values.

It seems likely that enterprises will have the lowest values of α_{sat}; at consumer web-services, compromise of one account has less potential to affect the whole network. As stated earlier, our focus is here on enterprise; nonetheless we suggest that damage probably grows faster than linearly at web-sites also. For example, online accounts at a bank should have minimal crossover effects, but at 25% compromise all confidence in the legitimacy of transaction requests and the privacy of customer data is likely lost.

For guessing attacks, the most easily guessed passwords fall first. Thus it is the weakest passwords that determine α_{sat}: the number of guesses that it takes to gather a cumulative fraction α_{sat} of accounts is what it takes to reach the saturation point. Since the attacker's ability to harm saturates once she reaches α_{sat}, the excess strength of the remaining (1-α_{sat}) of user passwords is wasted. For example, the strongest 50% of passwords might indeed be very strong, but from a system-wide viewpoint, that strength will only come into play when the other 50% of credentials have already been compromised (and the attacker already has the run of everything that is password-protected in the enterprise network).

In summary: password strength, or guessing resistance, is not an abstract quantity to be pursued for its own sake. Rather, it is a tool that we use to deny an attacker access to network resources. There is a saturation point, α_{sat}, where the network is so thoroughly penetrated that additional passwords gain the attacker very little; resistance to guessing beyond that point is wasted since it denies the attacker nothing. Thus, there's a "don't care" region after that saturation level of compromise, and for enterprise networks it appears quite reasonable to assume that α_{sat} is no higher than 0.1. At that level it is not simply the case that the weakest 10% of credentials are most important, but that the excess strength of the remaining 90% is largely irrelevant to the administrator's goal of system-wide defense. Of course there may be secondary

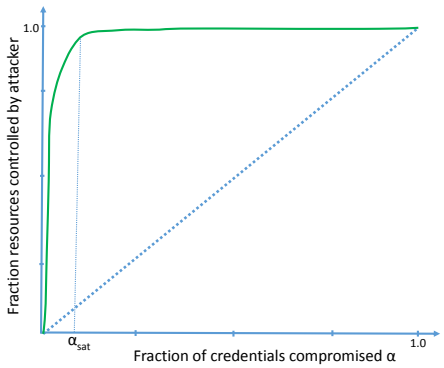

Figure 1: The fraction of network resources under attacker control grows faster than the fraction of credentials compromised. For example, given half of a system's credentials, an attacker likely effectively has access to all resources. For enterprise networks, we expect the saturation threshold, α_{sat}, where the network is completely penetrated, is likely under 0.1.

benefits to an individual user in having their password withstand guessing even after α_{sat} has been exceeded. For example, if re-used at another site the user still has a significant stake in seeing the password withstand guessing. Since our focus is on the administrator's goal of protecting a single site, this is not part of our model.

3 The online-offline chasm: too much and not enough

We next consider the difference between online and offline guessing. In online attacks an attacker checks guesses against the defender's server, *i.e.,* submitting guesses to the same server as legitimate users. In offline, she uses her own hardware resources, including networked or cloud resources and machines equipped with graphical processing units (GPUs) optimized for typical hash computations required to test candidate guesses. Thus offline attacks can test many orders of magnitude more guesses than online attacks, whether or not online attacks are rate-limited by system defenses. We consider online and offline guessing attacks separately.

An online attack is always possible against a web-facing service, assuming that it's easy to obtain or generate valid account userids. An offline attack is possible only if the attacker gains access to the file of password hashes (in order to verify correctness of guesses offline)—and in this case, an offline attack is necessary only if that file has been properly salted and hashed, otherwise simpler attacks are possible [6], e.g., rainbow tables for unsalted hashed passwords.

There is an enormous difference between the strength required to resist online and offline guessing attacks. Naturally, the probability of falling either to an online or offline attack decreases gradually with the number of guesses a password will withstand. A hundred guesses per account might be easy for an online attacker, but a thousand somewhat harder, and so on; at some point online guessing is no longer feasible. Similarly, at some point the risk from an offline attack begins to gradually decrease. Let T_0 be a threshold representing the *maximum* number of guesses expected from an online attack, and T_1 correspondingly the *minimum* number of guesses expected from a credible offline attack. (The asymmetry in these definitions is intentional, to provide a conservative estimate in reasoning about the size of the gap.) Then a password withstanding T_0 guesses is safe from online guessing attacks, while one that doesn't withstand T_1 guesses certainly won't survive offline attacks. Our own previous work [6] suggests $T_0 \approx 10^6$ and $T_1 \approx 10^{14}$ are reasonable coarse estimates, giving a gap eight orders of magnitude wide (see Fig.2); we emphasize however that the arguments herein are generic, regardless of the exact values of T_0 and T_1. While estimates for T_1 in particular are inexact, depending as they do on assumptions about attacker hardware and strategy (a previous estimate [6] assumed 4 month's of cracking against one million accounts using 1000 GPUs each doing a billion guesses per second), clearly T_1 vastly exceeds T_0.

A critical observation is that for a password P whose guessing-resistance falls between T_0 and T_1, incremental effort which fails to move it beyond T_1, is wasted in the sense that there is no guessing attack to which the original P falls, that the stronger password resists—since guessing attacks are either online or offline, with no continuum in between. Passwords in this online-offline gap are thus in a "don't care" region in which they do both too much and not enough: too much if the attack vector is online guessing, and not enough if it is offline. Once a password will withstand T_0 guesses, to stop additional attacks, any added strength must be sufficient to move to the right of T_1. While both online and offline attacks are countered by guessing-resistance, the amount needed varies enormously. In practical terms, distinct defences are required to stop offline and online attacks. This online-offline chasm then gives us a second "don't care" region, besides that defined by α_{sat}.

4 The "don't care" region: pushing on string

The compromise saturation point and the online-offline chasm each imply regions where there is no return on effort. The marginal return on effort is zero for improving any password which starts in the zone greater

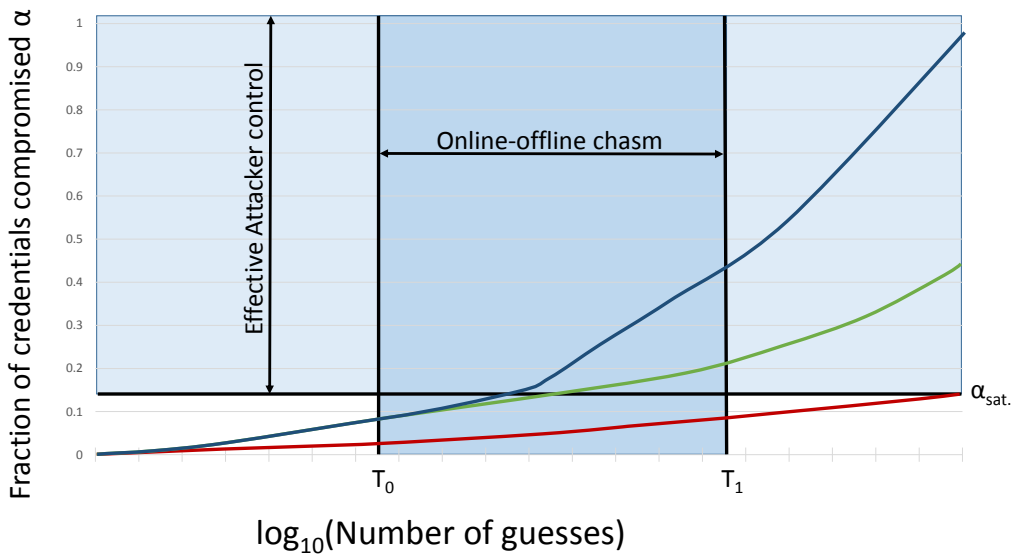

Figure 2: Don't care regions where there is no return for increasing effort. T_0 is the threshold above which online attacks cease to be a threat. T_1 is the threshold below which passwords almost surely will not survive credible offline attacks. α_{sat} is the threshold fraction of compromised accounts at which an attacker effectively has control of system resources. Examples for these parameters might be $T_0 = 10^6$, $T_1 = 10^{14}$, $\alpha_{\text{sat}} = 0.1$.

than T_0 yet remains less than T_1, and for passwords with guessing-resistance above the α_{sat} threshold. Fig.2 illustrates this with shaded areas denoting the "don't care" regions. A password withstanding $T_1 - \varepsilon$ guesses has the same survival properties as one surviving $T_0 + \varepsilon$—both survive online, but fall to offline attack, *i.e.,* equivalent outcomes. From the administrator's point of view, the strongest password in the population, *i.e.,* having the most guessing-resistance (the password at $\alpha = 1.0$) is similar to one at $\alpha_{\text{sat}} + \varepsilon$, in that both fall after the attacker's capability to harm is already saturated.

In summary, shaded regions denote these areas where there is no return-on-effort for "extra strength": 1) passwords whose guessing-resistance lies within the online-offline gap; and 2) passwords beyond where an attacker is gaining little from additional credentials. The size of the "don't care" region naturally depends on the particular values of α_{sat}, T_0 and T_1 but in all cases the shape of the password distribution (as defined by the colored curves in Fig.2) matters only in the areas below α_{sat} AND (left of T_0 OR right of T_1). An important observation is that, under reasonable assumptions, *the "don't care" regions cover a majority of the design space*. The relatively small unshaded regions are shown in Fig.2; outside of these regions changes to the password distribution accomplish nothing (at least from the administrator's viewpoint). To anchor the discussion, based on what we know of enterprise networks and attacker abilities, we've offered estimates of $\alpha_{\text{sat}} = 0.1, T_0 = 10^6$, and $T_1 = 10^{14}$, but choosing different values doesn't alter the conclusion that in large areas of the guess-resistance versus credentials-compromised space, changing the distribution of user-chosen passwords improves little of use; it causes no direct damage, but like pushing on string, is ineffective (and wasteful in energy).

To understand the consequences, Fig.2 also depicts guessing-resistance of three hypothetical password distributions. The blue (top) and green (middle) curves diverge widely on the right side of the figure—for a fixed number of guesses, far fewer green-curve accounts will be compromised than from the blue-curve distribution. The green-curve might appear better since those passwords are much more guess-resistant than those from the blue curve. Nonetheless they have identical attack survival outcomes since their divergence between T_0 and T_1 has minimal effect on performance against online or offline attacks, and divergence above α_{sat} happens only after the attacker's capacity for harm has already plateaued. The red (lower) curve shows a distribution that might survive an offline attack, as the curve lies below α_{sat} even at the number of guesses that an offline attacker might deliver. The enormous difficulty of getting users to choose passwords that will withstand offline guessing, and the waste that results unless almost all of them do has also been argued by Tippett [11]. We re-emphasize that α_{sat}, T_0 and T_1 are site and implementation dependent variables. We examine below how an administrator can vary them to decrease the size of the "don't care" zone.

5 What should an administrator optimize?

Ideally, a system's population of passwords would withstand both online and offline attacks. For this, the fraction of accounts compromised must be lower than α_{sat} at T_1 guesses (and ideally much lower, since an offline attacker may be able to go well beyond the expected minimum T_1). Unfortunately, recent work shows that user-chosen passwords don't even approach this, even when stringent composition policies are enforced. For example, Mazurek et al. [10] found that 48% of CMU passwords (which had a length-8 and 4-out-of-4 character sets policy in place) fell within 10^{14} guesses. On examining eight different password-creation policies, Kelley et al. [8] found that none kept the cumulative fraction of accounts compromised below 10% by 10^{11} guesses. Thus, if we believe that an attacker's control saturates by the time a fraction $\alpha_{\text{sat}} = 0.1$ of accounts is compromised, and that an offline attack can mount at least $T_1 = 10^{11}$ guesses per account, then there is little hope of resisting offline attack. That is, with these assumed values of α_{sat} and T_1, an attacker who gains access to the hashed password file will have all the access she needs, no matter how far outside her reach the remaining fraction (1-α_{sat}) of passwords lie.

What then should an organization seek to do? It is ideal if passwords robustly withstand both online and offline attacks, as is the case for the lower curve in Fig.2. However, striving for this but falling short wastes user effort and accomplishes nothing. In Fig.2, all effort above and beyond that necessary to withstand online attack is completely wasted in the two upper curves. An organization that tries to withstand offline attacks but fails to have at least a fraction $1-\alpha_{\text{sat}}$ of users survive T_1 guesses fares no better than one that never made the attempt (and does worse if we assume that user effort is a scarce resource [5]. The evidence of recent cracking work on real distributions [10, 8] suggests that this is the fate of more-or-less all organizations that allow user-chosen passwords (unless we believe that $\alpha_{\text{sat}} \approx 0.1$ is unreasonably low or $T_1 \approx 10^{14}$ is unreasonably high). Thus, the argument "stronger passwords are always better," while deeply ingrained, appears untrue. Stronger passwords lower the cumulative fraction of accounts compromised at a given number of guesses (*i.e.,* push the curves in Fig.2 lower). However, changes that occur within the shaded "don't care" regions happen when it no longer matters and do not improve outcomes.

It follows that a reasonable objective is to maximize, within reasonable costs, guessing-resistance across the system's set of passwords at the expected online and offline number of guesses, subject to the constraint that the fraction of compromised accounts stays below α_{sat}. That is, so long as $\alpha < \alpha_{\text{sat}}$, the lower α is at T_0 (resp. T_1) the better the resistance to online (resp. offline) attacks. For $\alpha > \alpha_{\text{sat}}$, improvements which do not reduce α below α_{sat} are unrewarded. Focussing as it does on a single site, we remind readers that the additional benefit of withstanding guessing to users who have re-used their password at other sites are not captured in this model.

6 What can an administrator control?

What tools does an administrator have to reach these goals? The outcome will be influenced by the values α_{sat}, T_0, and T_1 and the shape of the cumulative password distribution. We show these various forces in Fig.3.

The value of the compromise saturation point α_{sat} is largely determined by the network topology and might be relatively difficult to control or change in a given environment. Basic network hygiene and adherence to security principles (*e.g.,* principles of least privilege, isolation and containment), can help minimize damage when intrusion occurs. Of course, these defenses are also effective against intrusions that do not involve password guessing. We assume that these defenses will already be in force, and in the rest of the section concentrate on measures that mostly affect password guessing attacks.

Improving T_0: There aren't many reasons for any authentication system to allow hundreds of thousands of distinct guesses on a single account. In cases of actual password forgetting it is unlikely that the legitimate user types more than a dozen or so distinct guesses. Mechanisms that limit the number of online guesses (thus reducing T_0) include various throttling mechanisms (rate-limiting) and IP address blacklisting. The possibility of denial of service attacks can usually be dealt with by IP address whitelisting. (We mean, not applying the throttling triggered by new IP addresses to known addresses from which a previous login succeeded; wrong guesses from that known address should still be subject to throttling.) A simple easily-implemented throttling mechanism may suffice for many sites. When denial of service attacks are a possibility, more complex mechanisms may be necessary, perhaps including IP address white- and blacklisting, and methods requiring higher effort from site administrators. Such defensive improvements should come without additional burdens of effort and inconvenience to users. Together with password blacklisting (discussed below), throttling may almost completely shut down generic online guessing attacks.

Improving T_1: A password must withstand a certain number of guesses to have any hope of withstanding credible offline attacks. A lower bound T_1 on this number may vary depending on the defenses put in place, and can be very high. For example, if an attacker can make ten billion guesses per second on each of one thousand GPUs [7] then in a four month period he can try about

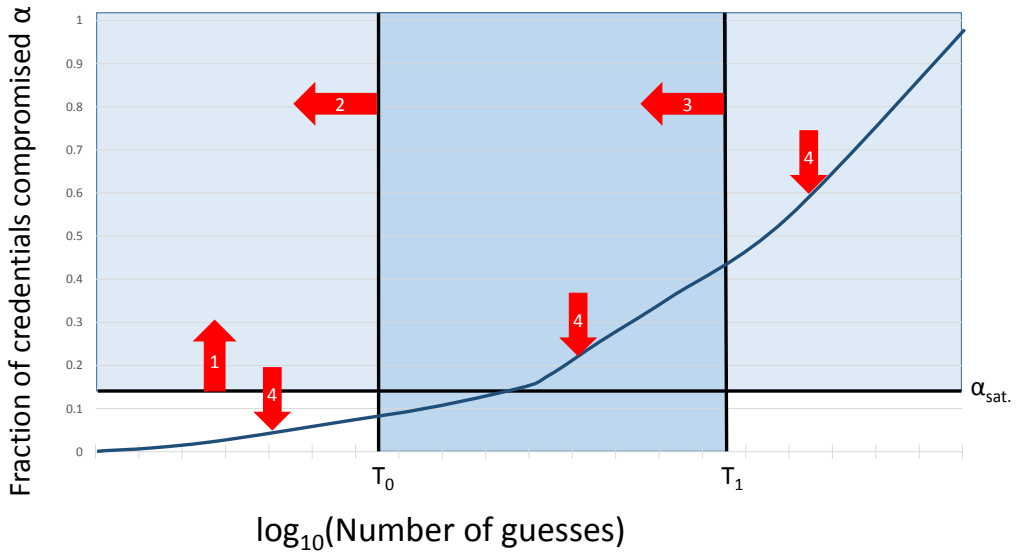

Figure 3: Defensive elements aiming to improve guessing-resistence. (1) α_{sat} (the point at which attacker control saturates) can be raised by implementation of basic security principles such as least-privilege and compartmentalization. (2) T_0 (the maximum number of online guesses) can be reduced by throttling mechanisms. (3) T_1 (the minimum number to be expected from an offline attack) can be reduced by iterated hash functions. (4) The cumulative fraction of accounts that have fallen at a given number of attacker guesses can be reduced (pushing the blue curve down) by improving the guessing-resistance of user-chosen passwords, *e.g.,* to the left of T_0 by password blacklisting, and by password composition policies generally. Changing α_{sat}, T_0 and T_1 alter the size of shaded "don't care" regions.

$T_1 = 10^{14}$ guesses against each of one million accounts [6]. Furthermore, technology advances typically aid attackers more than defenders; few administrators will replace hardware every year, while attackers can be assumed to have access to the latest resources. This moves T_1 to the right—as computing speeds and technology advance, and customized hardware gives attackers further advantages. Since the hardware is controlled by the attacker, little can be done to directly throttle offline guessing. However, an effective way to reduce the number of trials per second is to use a slow hash, i.e., one that consumes more computation. The most obvious means is hash iteration [12]; recent research is also exploring the design of hash functions specifically designed to be GPU-unfriendly.

For example, ignoring the counter-acting force of speed gains due to advancing technology, an iteration count of $n = 10^4$ reduces T_1 from 10^{14} to 10^{10}. Even with iteration it is hard to move T_1 all the way left to T_0 to make the online-offline chasm disappear. Limits on further increasing n arise from the requirement that the time to verify legitimate users must be tolerable to both the users (wait time) and system hardware; *e.g., n* might allow verifying 100 legitimate users/sec (10ms per user). If 10ms is a tolerable delay an attacker with access to 1000 GPUs can compute a total of $1000 \times 4 \times 30 \times 24 \times 60 \times 60/10^{-2} \approx 10^{12}$ guesses in 4 months. Directing this effort at 100 accounts would mean that each would have to withstand a minimum of $T_1 = 10^{10}$ guesses. Since these are conservative assumptions, it appears challenging to decrease T_1 below this point.

Note that, as technology evolves, the number of hash iterations can easily be increased, invisibly to users and on-the-fly—by updating the stored password hashes in the system-side file to reflect new iteration counts [12]. Among the appealing aspects of iterated hashing, it is long-known as an effective defensive tool, and costs are borne system-side rather than by user effort. However, hash iteration is not a miracle cure-all—for a password whose guess-resistance is 10^6, online throttling is still important; an online attacker who could test one password every 10ms (matching the system rate noted above) will succeed in 10^4s = 2hr 47min.

Eliminating offline attacks altogether: The emphasis on encouraging users to choose stronger passwords can obscure the fact that offline attacks are only a risk when the password hash file "leaks" (a euphemism for "is somehow stolen") or otherwise becomes available to an attacker. Any means or mechanisms that prevent the file from leaking *entirely remove* the need for individual passwords to withstand an offline attack. Since we de-

fined T_1 as the minimum number of guesses a password must withstand to resist an offline attack, any such mechanism effectively reduces T_1 to zero. One particularly appealing such mechanism is discussed next.

HARDWARE SECURITY MODULES (HSMS). A properly used HSM eliminates the risk of the hash file leaking [6], or equivalently eliminates the risk of a decryption key (or backup thereof) leaking in the case that the means used to protect the information stored system-side to verify passwords is reversible encryption. In such a proper HSM architecture, rather than the one-way hash of salted passwords, what is stored is a message authentication code (MAC) computed over the password using a secret key. When a password candidate is presented for verification, the candidate plus the corresponding MAC from the system file are provided as HSM inputs. The HSM holds the system secret key used to compute the MAC; importantly, this secret key is by design never available outside the HSM. Upon receiving the (MAC, candidate password) input pair, the HSM independently computes a MAC over the input candidate, compares it to the input MAC, and answers yes (if they agree) or no. Stealing the password hash file—in this case a password MAC file—is now useless to the offline attacker, because the HSM is needed to verify guesses. In other words, offline attacks are no longer possible.

Another interesting scheme to mitigate offline attacks was proposed by Crescenzo et al. [2]. It bounds the number of guesses that can be made by restricting the bandwidth of the connection between the authentication server and a specially constructed hash-server (which requires a very large collection of random bits). This limits an attacker to online guessing since, by design, the connection is too small to support the typical guessing rate an offline attacker needs, or to allow export of any file that would be useful to an offline attacker.

Improving the password distribution: Finally, we consider changes to the password distribution as a means of improving outcomes. Recall that the curves in Fig.2 represent the cumulative fraction of accounts compromised as a function of the number of guesses-per-account. In general it has been thought that the lower this cumulative fraction the better; a great deal of effort has gone into coercing users to choose "stronger" passwords (thus pushing the cumulative distribution curve downward, in one or more of the three regions induced by T_0 and T_1). However, as has been explained, lower is of tangible benefit only outside of the "don't care" region: improvements to the curve inside the "don't care" region have negligible effect on outcomes in any attack scenario.

First, note that tools to influence the cumulative distribution are mostly indirect: it is users who choose passwords, not administrators. For example, by some combination of education campaigns, password policies and password meters, administrators may try to influence this curve towards "better" passwords. However, the cumulative distribution is ultimately determined by user password choice; if users ignore advice, do the minimum to comply with policies, or aren't motivated by meters, then efforts to lower the curve may have little impact.

Second, note that many policy and education mechanisms are unfocused, in the sense that they cannot be targeted at the specific part of the cumulative distribution where they make most difference (and away from the "don't care" region where they make none). Even if they succeed, exhortations to "choose better passwords" aren't concentrated at one part of the curve or another; if all users respond to such a request by improving their passwords marginally, the related effort of 90% of users is still wasted for an enterprise where $\alpha_{\text{sat}} = 0.1$. We now examine common approaches to influencing password choices in this light.

PASSWORD BLACKLISTING. Attackers exploit the fact that certain passwords are common. Thus, explicitly forbidding passwords known to be common can reduce risk. Blacklists concentrate at the head of the distribution, blocking the choice of most common passwords. For example, a user who attempts to choose "abcdefg" is informed that this is not allowed and is asked to choose another. Certain large services do this; Twitter blacklisted 380 common passwords after an online guessing incident in 2009, and Microsoft applied a blacklist of several hundred to its online consumer properties in 2011. With improvements of password crackers, and the recent wide availability of passwords lists, blacklists need to be longer.

A blacklist of, say 10^6 common passwords, may help bringing guessing resistance to the 10^5 level. A natural concern with blacklists is that users may not understand why particular choices are forbidden. Kelley et al. [8] examine the guessing resistance of blacklists of various sizes, but the question of how long one can be before the decisions appear capricious is an open one. Komanduri et al. [9] pursue this question with a meter that displays, as a user types, the most likely password completion. A further unknown is the improvement achieved when users are told their password choice is forbidden. It appears statistically unlikely that all of the users who initially selected one of Twitter's 380 blacklisted passwords would collide again on so small a list when making their second choice, but we are not aware of any measurements of the dispersion achieved. A promising recent practical password strength estimator is `zxcvbn` [15]; it can also be used to help blacklisting.

Observe that blacklists are both direct and focused: they explicitly prevent choices known to be bad rather than relying on indirect measures, and they target those

making bad choices, leaving the rest of the population unaffected. Using a blacklist appears one of the simplest measures to meaningfully improve the distribution in resisting online attacks.

COMPOSITION POLICIES. Composition policies attempt to influence user-chosen passwords by mandating a minimum length and the inclusion of certain character types—a typical example being "length-8 and use three of the four character sets {lowercase, uppercase, digits, special characters}". Certain policies may help improve guess-resistance in the 10^5 to 10^8 range. However, for what we have suggested as the reasonable values $\alpha_{\text{sat}} = 0.1$ and $T_1 = 10^{14}$, the evidence strongly suggests that none of the password composition policies in common use or seriously proposed [10, 8] can help; for these to become relevant, one must assume that an attacker's ability to harm saturates much higher than $\alpha_{\text{sat}} = 0.1$, or that he can manage far fewer than $T_1 = 10^{14}$ offline guesses. Thus, such policies fail to prevent total penetration of the enterprise network. Their ineffectiveness is perhaps the reason why a majority of large web services avoid onerous policies [4].

Note that composition policies are indirect: the constraints they impose are not themselves the true end objectives, but it is hoped that they result in a more defensive password distribution. This problem is compounded by the fact that whether the desired improvement is achieved is concealed from an administrator: the justifiably recommended practice of storing passwords as salted hashes means that the password distribution is obscured, as well as any improvements caused by policies. Composition policies are also unfocused in that they affect all users rather than being directed specifically where they may matter most. A policy may greatly impact user password choice and still have little impact on outcome—e.g., if all of the change in the cumulative distribution happens inside the "don't care" region.

7 Concluding remarks

In summary, password strength, which actually means guessing-resistance, is not a universal good to be pursued for its inherent benefits—it is useful only to the extent that it denies things to adversaries. When we consider a population of accounts there are large areas where increased guessing-resistance accomplishes nothing—either because passwords fall between the online and offline thresholds, or because so many accounts have already fallen that attacker control has already saturated. If increases in password guessing-resistance were free this would not matter, but they are typically achieved at great cost in user effort—for example, there is a void of evidence that current approaches based on password composition policies significantly improve defensive outcomes, and strong arguments that they waste much user effort. This thus creates risk of a false sense of security.

It is common to assume that users must choose passwords that will withstand credible offline attacks. However, if we assume that an offline attacker can mount $T_1 = 10^{14}$ guesses per account, and has all the access she needs by the time she compromises a fraction $\alpha_{\text{sat}} = 0.1$ of accounts, we must acknowledge that trying to stop offline attacks by aiming user effort towards choosing "better passwords" is unachievable in practice. The composition policies in current use seem so far from reaching this target that their use appears misguided. This is not to say that offline attacks are not a serious threat. However, it appears that enterprises that impose stringent password composition policies on their users suffer the same fate as those that do not. If the hashed password file leaks, burdening users with complex composition policies doesn't alter the fact that a competent attacker likely gets access to all the accounts she could possibly need. Nudging users in the "don't care" region (where most passwords appear to lie) is simply waste.

The best investments to defend against offline attacks appear to involve measures transparent to users. Iteration of password hashes lowers the T_1 boundary; however, even with very aggressive iteration we expect that at least 10^{10} offline guesses remain quite feasible for attackers. Use of message authentication codes (MACs), i.e., keyed hash functions, instead of regular (unkeyed) password hashes, provides effective defence against offline attacks which exploit leaked hash files—provided that the symmetric MAC key is not also leaked. One method of protecting MAC keys is hardware security modules (HSMs) as discussed—though more expensive than software-only defences, they can entirely eliminate offline attacks. Online guessing attacks, in contrast, cannot be entirely eliminated, but effective defences include password blacklists and throttling—and there appear few barriers to implementing these simple defences.

Acknowledgements. We thank Dan Wheeler and anonymous referees for comments which improved this paper. The third author is Canada Research Chair in Authentication and Software Security and acknowledges NSERC for funding this chair and a Discovery Grant.

References

[1] P. Bright. RSA finally comes clean: SecurID is compromised. Ars Technica, June 2011. `http://arstechnica.com/security/news/2011/06/rsa-finally-comes-clean-securid-is-compromised.ars/`.

[2] G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In 2006 Theory of Cryptography Conference, pages 225–244.

[3] Dunagan, J. and A. X. Zheng and D. R. Simon. Heat-ray: combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs. 2009 ACM Symp. on Operating Systems Principles, pages 305–320.

[4] D. Florêncio and C. Herley. Where do security policies come from? SOUPS 2010.

[5] D. Florêncio, C. Herley, and P. van Oorschot. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. USENIX Security 2014, pages 575–590.

[6] Florêncio, D. and Herley, C. and van Oorschot, P.C. An Administrator's Guide to Internet Password Research. USENIX LISA 2014, pages 35–52.

[7] D. Goodin. Why passwords have never been weaker and crackers have never been stronger. Ars Technia, 2012. `http://arstechnica.com/security/2012/08/passwords-under-assault/`.

[8] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. 2012 IEEE Symp. Security and Privacy, pages 523–537.

[9] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. Schechter. Telepathwords: Preventing weak passwords by reading users minds. USENIX Security 2014, pages 591–606.

[10] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. Kelley, R. Shay, and B. Ur. Measuring password guessability for an entire university. ACM CCS 2013.

[11] P. Tippett. Stronger Passwords Aren't. Information Security Magazine, pages 42–43, June, 2001.

[12] N. Provos and D. Mazières. A future-adaptable password scheme. 1999 USENIX Annual Technical Conference: FREENIX track, pages 81–91.

[13] RSA FraudAction Research Labs. Anatomy of a hack. April 1, 2011, `https://blogs.rsa.com/anatomy-of-an-attack/`.

[14] B. Toxen. The NSA and Snowden: Securing the All-Seeing Eye. Comm. ACM, 57(5):44–51, May 2014.

[15] D. Wheeler. zxcvbn: Low-budget password strength estimation. USENIX Security 2016.