Quantifying the Security Advantage of Password Expiration Policies*

Sonia Chiasson and Paul C. van Oorschot
School of Computer Science, Carleton University, Ottawa, Canada

Abstract. Many security policies force users to change passwords within fixed intervals, with the apparent justification that this improves overall security. However, the implied security benefit has never been explicitly quantified. In this note, we quantify the security advantage of a password expiration policy, finding that the optimal benefit is relatively minor at best, and questionable in light of overall costs.

1 Introduction and model

Password aging policies, also called *password expiration* policies, force users to change passwords within fixed intervals, e.g., every 30 days or six months. The historical idea [2, 4, 10] is that this increases security—although the implied gain has never been quantified. Nonetheless, password expiration policies remain common in practice [8]. In this note, for the first time, we explicitly quantify the security gain of changing passwords under an appropriate analytic model, relative to an ongoing guessing attack.

Password expiration aims to either decrease the chances of an adversary coming into possession of an account password, or to respond to it—however the effectiveness of the latter is seriously called into question by research showing (see Section 4) that when password changes are forced, often new passwords are algorithmically related to old, allowing many to be found in few guesses. This presumably leaves the main benefit to be from decreasing the chances of a password being guessed while it remains active.

The practice of password expiration was widely motivated [3] by guessing attacks. Historically, offline guessing attacks have been cited as a major concern [4], albeit requiring a relatively limiting set of circumstances [9] including possession of verifiable text against which to test guesses offline, e.g., password hashes from a compromised password file. Online attacks allow fewer guesses, but are easily mounted.

To facilitate analysis, our attack model is as follows. An attacker is in the process of exhaustively guessing passwords for a given account. For simplicity, assume online guessing (though we also discuss offline guessing). The guessing attack continues in parallel with any user password changes within the expiration policy period. We assume the attacker knows the length of the policy period, but not the precise time the user changes passwords. The attack strategy is a deterministic search, for a password contained in a known, finite space; note that this implies that an exhaustive search is guaranteed to eventually succeed in the absence of a password change. The attacker, knowing that a change policy is in effect, whenever reaching the end of the password search space without success will begin searching the space anew (possibly in a different deterministic order).

We ask: what defensive advantage results from a password change conforming to the policy? Section 2 provides the base analysis for the simpler problem of cryptographic key search with randomly chosen keys. Section 3 contextualizes the results for the main problem of interest: guessing user-chosen passwords. Section 4 notes related work. Section 5 concludes.

2 Exhaustive search with changing keys

We first model password guessing as an exhaustive key-search problem, using as a framework the related cryptanalytic question: how does changing a cipher key affect an attacker's probability of correctly guessing that key? The attacker is assumed to have suitable plaintext-ciphertext pairs for testing purposes. This problem is harder for the attacker (i.e., takes longer before success) due to a random-key assumption, but simpler to analyze than that of guessing user-chosen passwords; we later explain the relationship.

*March 17, 2015. For author's personal use. To appear: *Designs, Codes and Cryptography*. The final publication is available at Springer via http://dx.doi.org/10.1007/s10623-015-0071-9

Suppose message M is encrypted to ciphertext $C = E_k(M)$ under cipher E and key $k \in \{k_1, k_2, \ldots, k_R\}$, where we assume $R = 2^r$ equi-probable keys in the space. An attacker V tries to guess the correct key by proceeding one guess at a time exhaustively through some arbitrary but deterministic ordering of the key space; k_i is the i^{th} key in this ordering. For context, we first consider some simple questions of interest.

Q1: What is V's probability of success for a single guess? Answer: $1/R$.

Q2: What is V's probability of success after c guesses? Answer: c/R.

Q3: What is V's expected number N of guesses before success? Answer: $N = R/2$.

Thus if the time required by V to exhaustively test the full set of keys is T, we expect a successful guess halfway through the search, at time $T/2$.

2.1 Towards the problem of interest

To approach our problem of interest, the game is changed as follows. V guesses k_i at time t_i, $i \geq 1$. At a random time $t_{u+1} \in [t_2, t_R]$ unknown to V, the key under which M is encrypted is changed to a random key k^*. The correctness of key guesses thereafter is relative to k^*. Assume V completes the original key search sequence, guessing each of the R keys once ordered by some random permutation. (An oracle answers whether each guessed key matches the active key; this is analogous to an online password guessing attack.) Our main question is: does changing k to k^* give a security advantage, reducing the probability of successful attack in a fixed period? The answer is yes, as explained next—but the advantage is small. (If you find this counter-intuitive, consider this question, related to Q1 above: if z is the probability that an attacker's next guess is your password, how does z change if you change your password just before the guess?)

2.2 The base case $T \leq P$

NOTATION. Let p_s denote the probability of attack success over R exhaustive guesses; T the number of units of time the attacker needs to test the full set of R guesses; and P the password policy expiration period.

For $T \leq P$, determining p_s involves considering two time intervals: $[t_1, t_u]$ with k active, and $[t_{u+1}, t_R]$ with k^* active; $u \geq 1$. Success means finding either key while it is active. We visualize a timeline from t_1 to t_R with k_i guessed at time t_i. This partitions the keyspace into $W_1 = \{k_1, \ldots, k_u\}$ and $W_2 = \{k_{u+1}, \ldots, k_R\}$. Let $q = u/R$. Table 1 details the four cases to consider. From it, the overall probability of success is $p_s = q_1 + q_2 + q_3 = 1 + q^2 - q$. The probability of failure, $p_f = q - q^2$, has maximum 0.25 at $q = 1/2$ (i.e., at $u = R/2$) occurring when time is split evenly over the keys. Thus the defender's best strategy is to change keys at exactly the mid-point of the time T it takes to search the entire keyspace; of course, it is also reasonable to expect users to delay a change until forced as P expires. Also, since real attackers do not inform defenders of their T value, even motivated defenders are unlikely to ever attain this best-case reduction, from 1.0 to 0.75, of attacker success probability over R guesses.

Case	Events	Result	Probability
1	$k \in W_1, k^* \in W_2$	success	$q_1 = (q)(1-q)$
2	$k \in W_1, k^* \notin W_2$	success	$q_2 = (q)(q)$
3	$k \notin W_1, k^* \in W_2$	success	$q_3 = (1-q)(1-q)$
4	$k \notin W_1, k^* \notin W_2$	failure	$q_4 = (1-q)(q)$

Table 1: **Key search outcomes** ($q = u/R$; see text).

Smaller values T (i.e., faster search) decrease the probability of a key change (cf. user password change) occurring before a successful guess, moving the probability of attacker success p_s per search period T closer to 1.0. As T approaches P from below, the probability of a key change within T increases.

The attacker improves his probability of success beyond $p_s^* = 0.75$ by starting a fresh search on failure after R guesses. The probability $p_s^{(i)}$ of success within i search periods T (time $i \cdot T$) is 1 minus the probability of failure on all i searches. Thus for $p_f^* \leq 0.25$ and assuming independence, $p_s^{(i)} = 1 - (p_f^*)^i \geq 1 - 4^{-i}$. Attack success is near certain even for small i; already for $i = 4$ search periods T we have $p_s^{(4)} \geq 0.996$.

2.3 The case $T > P$

Next, consider $T > P$. If $T = 2P$, and the user delays a key change until forced at time P, then the change for an individual user is at the mid-point of the search period T. With p_s replaced by $p_{s,t}$ now for $T = t \cdot P$, this achieves $p_{s,2} = 0.75$, the minimum (i.e., best case for defender) under the base analysis. Unfortunately, the value p_s is per exhaustive search period T; by starting a new search on failure, even unlucky attackers expect success within just a few periods T, as noted above.

To generalize this for $T = t \cdot P$, first consider $t = 3$. Envision a timeline of length $3P$ partitioning the search space into intervals W_1, W_2, W_3 (cf. base case above). Assume a user delays changing their key (password) until policy requires. Then each W_i has "length" corresponding to $R/3$ keys, i.e., 1/3 of the keyspace each. Let $k^{(i)}$ denote the user key active during interval W_i. For the attack to succeed, the attacker must guess at least one $k^{(i)}$ while that key is active; the attack fails if and only

if $(k^{(1)} \notin W_1) \wedge (k^{(2)} \notin W_2) \wedge (k^{(3)} \notin W_3)$. The fraction of candidates that an attacker can try in each interval is 1/3. With the model assuming random choices and search order (equi-probable keys), the probability for each individual event $k^{(i)} \notin W_i$ is $(1-(1/3))$. Thus the probability the attack fails is $p_f = p_{f,3} = (1-(1/3))^3 \approx 0.296$.

For $T = t \cdot P$ the above generalizes to give a main result of

$$p_{f,t} = (1 - 1/t)^t$$

with limit as $t \to \infty$ of $p_{f,\infty} = 1/e \approx 0.368$, and thus probability of attack success $p_{s,\infty} \approx 0.632$ over a single exhaustion period T. As an attacker's search power weakens (i.e., larger T to search full space), the expectation of attack success should fall; but to repeat, this analysis tells us that for $T = t \cdot P$ in the limit (as $t \to \infty$), the attack success expectation drops to no lower than $p_{s,\infty} \approx 0.632$, from the $t = 2$ lower bound $p_{s,2} = 0.75$.

Restarting a search upon failure after completing an initial search period T again improves the attack success probability, as noted earlier. By executing i successive searches, success probability over time $i \cdot T$ improves to

$$p_{s,t}^{(i)} = 1 - (p_{f,t})^i = 1 - (1 - 1/t)^{it}.$$

For example, at $t = 10$ and $i = 4$ we have $p_{s,10}^{(4)} \approx 0.985$.

REMARK ON $T \gg P$ (MANY KEY CHANGES WITHIN PERIOD T). The results for case $T > P$ above naturally apply for subcase $T \gg P$, wherein key changes would occur many times before the attack completes a single full search cycle. Note that in terms of attack success, more important than the number of changes is the attack time needed to cover the search space (or a subset in which the key is expected to be with high probability). As Section 4 discusses, from Desmedt [6] we already know that for an analogous key search problem, the probability of attacker success remains high even if the key is changed after every guess.

REMARK ON OFFLINE ATTACK SPEEDS VS. EXPIRATION PERIODS. We comment here on the relevance of password aging in light of the speed of offline attacks. Originally, an explicit goal of password expiration policies was to bound the risk of falling to a year-long attack to one in a million (see [4])—in essence, asking for the condition $T \gg P$. With respect to offline attacks today, this goal is unattainable in the face of modern resources which easily allow 7-10 billion guessing trials per second [4, 9]; even if users chose 8-character passwords totally at random from a set of 93 symbols, exhaustive search on the full space of $93^8 = 2^{52.3}$ elements takes only 9.2 days, and the explicit goal would require users change passwords every 800 milliseconds (clearly nonsense).

This leads us to revisit the relationship between T and P in today's environment. One reality is very fast offline search capabilities; T is getting smaller. Another is the necessity to keep expiration periods P tolerable to users—decreasing P appears unacceptable. If full offline searches are completed in time T before a key (password) is changed, then *the aging policy provided little protection against the offline guessing attack.* Since offline attacks are so much faster, this leads us to return our analysis focus to online attacks, to see what protection is possible there. In practice user-chosen passwords are not random and skewed real-world password distributions allow online guessing attacks with improved efficiency, as discussed next.

3 Relationship to Password Guessing with Aging Policies

The analysis above holds for equi-probable passwords—e.g., as might arise from system-assigned passwords. However, such systems in which passwords are sufficiently user-friendly remain elusive. From our analysis, one might conclude that even for idealized systems, the security gain delivered by password aging is small relative to extra burdens introduced. As we now consider, for password distributions as found in practice, the situation is worse in the following sense: knowledgeable attackers can expect success earlier (though the success probability over full space searches remains as detailed above).

In moving from analyzing the impact on security of password aging on idealized passwords to guessing user-chosen passwords in the real world, the analysis is more involved due to password length variation and unknown skewed distributions. The first is easily handled: though in some cases users may choose passwords of unconstrained length, the vast majority fall within well-defined bounds. This justifies modelling finite spaces of n-character passwords for fixed n (e.g., $n \leq 8$ or 12), yielding reasonable approximations.

The second issue is less tidy. As is well-known, user-chosen passwords are far from equi-probable; skewed distributions result as password choices of many users follow predictable patterns (see Section 4). Knowledge on optimizing practical online guessing attacks, and improved metrics for measuring their efficacy, has been advanced by Bonneau [1] in conjunction with his (privacy-preserving) study of the largest natural dataset to date, and, e.g., analysis by Weir et al. [17] of large datasets of plaintext passwords available due to publicized compromises. A few highlights help us contextualize.

In practice, optimal adversaries focus effort on the easiest targets first, and are "early quitters" in the sense

of rarely fully exhausting a guessing space. The optimal attack tries passwords in decreasing order of probability.[1] This is effectively measured by *partial guessing metrics* [1] such as the β-success rate giving an attacker's probability of success after β guesses:[2] $\lambda_\beta(\chi) = \sum_{i=1}^{\beta} p_i$, where p_i is the probability of password x_i from distribution χ, with p_i in decreasing order. Trying higher-probability candidates first naturally results in expected success earlier. Skewed distributions thus decrease the attack work by optimal adversaries.

In scenarios where exhaustive search is abandoned before completion, "earlier" may mean success versus failure. In the context of the §2 analysis, an attacker guessing in probability order on passwords from skewed distributions can expect to succeed earlier in a period T. However as noted, the expectation of success over the full exhaustive period T remains per the analysis—as it measures success relative to a complete search over time period T. While we avoid herein possibly contentious assumptions about specific probability distributions, we start §4 with examples illustrating how surprisingly effective optimal searches have been on specific datasets.

In summary, whether considering idealized equiprobable passwords per the Section 2 model (which is best-case for defenders), or the practical reality of skewed distributions noted here, the maximum advantage that a defender can hope to gain by a policy-driven password change is a reduction in the expectation of attack success over a single period T, from 1.0 (guaranteed success over the full period) to a probability no lower than (and as discussed, likely higher than) 0.75 for the case $T \leq P$ and in no cases any lower than 0.632 per period T under any scenario $T > P$ discussed. The attacker has yet further opportunities for success by starting fresh searches in subsequent periods T (or, at his option, restarting at any point after a sufficient fraction of high-probability candidate passwords is tried within any given period T).

4 Related Work

Curry [5, p.20] notes that password aging is a long-known defensive mechanism (cf. [2, 4]). Weir et al. [17] found, as part of statistical analysis of real-world password datasets including one of over 32 million passwords, that the most popular 50,000 items from a training sub-list of 5 million cover over 25% of passwords (when tested on a disjoint sub-list of 1 million passwords from the original dataset) of character-length 7 or more, and 14% of those length 10 or more. Bonneau [1], from analysis of a natural dataset of 70 million passwords, estimated that an online guessing attack trying the most-popular few passwords on each of a large number of accounts, say 10 per account, yields about 1% of passwords (thus relative to this particular measure, passwords offer the security of about 10-bit random strings); and an optimal attacker able to execute a massive search can find about 50% of all account passwords after about 1 million guesses per account (so by this measure passwords roughly equate to 20-bit random strings). Thus for skewed password distributions as occur in practice, online attacks require relatively few guesses to be damaging.

Zhang et al. [18], in a 2010 empirical study, found that for a large proportion of user accounts at a major U.S. university, knowledge of an existing password allowed a user-chosen next password to be predicted with high success using heuristic algorithms. For example, they successfully guessed passwords for 41% of users in a dataset of 7700 accounts in an offline attack with expected effort of a few seconds using one machine, and broke 17% of accounts on average in fewer than 5 online guesses in expectation. These results enjoy a filtering bias: the dataset consisted of the 76% of passwords recoverable by cracking tools, thus corresponding to the "easiest" passwords.

Both anecdotal reports and evidence-based research (e.g., Weir et al. [17]; see above) indicate that subsets of users tend to choose passwords which minimally satisfy password composition policies. Mazurek et al. [12] mention the concern that users who find composition policies "annoying" may comply with policy in predictable ways. Skewed distributions are targeted by heuristic password-guessing tools [11, 13, 16]. It is now understood [1, 17] that guessing-resistance for passwords should be measured by partial guessing metrics (see §3), not entropy-based metrics [3]. Password compromise is one application modelled by a variant of the FlipIt game of van Dijk et al. [7], who pursue a game-theoretic analysis.

Desmedt [6, pp.50-51] considered an exhaustive deterministic cryptanalysis machine M searching for a crypto key, with T the time required for a full exhaustive (deterministic) search. If the key is changed after every S seconds, and the attacker both restarts M at a random starting point upon each such change (obtaining a signal of the exact change times) and gains access to a plaintext-ciphertext pair under each new key, a successful key-finding attack remains possible; even changing the key arbitrarily frequently (in the limit, for each message transmitted) does not prevent successful attack.[3]

[1] These probabilities are unknown and change across datasets; estimates are used, based on large datasets accumulated from prior compromises, or from heuristic tools.

[2] More precisely, this is for β guesses per account. The optimal attack tries the most probable password on each account, then the next most probable, etc.

[3] If this is counter-intuitive, note that an attack which guesses key candidates in a fixed sequence actually benefits from a key change if the original target key is more distant in the guessing sequence than the

The probability of finding a correct key within time $i \cdot T, i \geq 1$, becomes (in the limit) $1 - 1/e^i \approx 0.632$ for $i = 1$. Our complementary analysis herein began by modelling a key being changed once at an arbitrary point within the full search period T and assuming the attacker continues in a sequential deterministic search (vs. changing keys arbitrarily often and restarting the key search equally often); we then generalized to cases where over the time $T = t \cdot P$ for one exhaustive search, the number of aging policy periods P is $t = 2, 3, ...$ with user key (password) changes at the end of each period P.

In the context of distributed computations involving a large number of independent machines, Quisquater and Desmedt [14, p.18] consider the difference in effectiveness between exhaustive (deterministic) key search machines and exhaustive (random) key search machines, where the former proceed sequentially through arbitrary but deterministic permutations of the key space, while the latter test random keys. They show that for T as defined above, the expected time to success for a randomized search is T, twice the time expected for success in a deterministic search (which as usual is $T/2$, with success expected half-way through the sequence, and guaranteed upon full completion at time T).

5 Concluding Remarks

A forced password change stops ongoing account access by adversaries who have come into possession of an account password (including friends given a password "temporarily"), who may otherwise enjoy continued access through that password. However, it provides little help against numerous other attacks, including those which upon first access immediately procure target files, set up a back door, or install keystroke-logging software or other persistent malware to render ineffective subsequent password changes. For example, in many operating systems, the "change password" command is easily redefined within a user environment (i.e., without privilege escalation) to execute that system function followed by sending the new password to a colluding site, e.g., by browser through a URL query parameter, or by email. In sum, these security-specific observations and the results in Section 3 suggest the security benefit of password aging policies are at best partial and minor. Combining this with the well-known and widely experienced (negative) usability impact of password aging policies, and results [18] mentioned earlier on high predictability of new passwords from knowledge of old, the burden appears to shift to those who continue to support password aging policies, to explain why, and in which specific circumstances, a substantiating benefit is evident.

newly updated key. In our analogous problem herein, the implication is that a successful guessing attack cannot be prevented even if a user changes passwords continuously, as quickly as system interfaces allow.

Acknowledgements. We thank Joseph Bonneau and anonymous referees for insightful comments which have improved this paper. Both authors acknowledge funding from Canada's NSERC for Canada Research Chair and Discovery Grant funding.

References

[1] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. IEEE 2012 Symposium on Security and Privacy.

[2] J. Bonneau, C. Herley, P.C. van Oorschot, F. Stajano. The past, present, and future of password-based authentication on the web. *Comm. ACM* (to appear, 2015).

[3] W. Burr, D.F. Dodson, W.T. Polk (editors). Electronic Authentication Guideline. NIST Special Pub 800-63 Version 1.0, June 2004. (Later versions include Burr et al., NIST SP - 800-63-2, August 2013.)

[4] W. Cheswick. Rethinking passwords. *Comm. ACM* 56(2):40–44, February 2013.

[5] D.A. Curry. *UNIX System Security: A Guide for Users and System Administrators.* Addison-Wesley, 1992.

[6] Y. Desmedt. Unconditionally secure authentication schemes and practical and theoretical consequences. Crypto'85, pp.42-55, LNCS vol.218, Springer-Verlag, 1986.

[7] M. van Dijk, A. Juels, A. Oprea, R.L. Rivest. FlipIt: the game of "stealthy takeover". J. Cryptology 26(4): 655-713 (2013).

[8] D. Florencio, C. Herley. Where do security policies come from? ACM SOUPS 2010.

[9] D. Florencio, C. Herley, P.C. van Oorschot. An administrator's guide to Internet password research. USENIX LISA 2014.

[10] C. Herley, P.C. van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy* 10(1):28-36 (Jan/Feb 2012).

[11] P.G. Kelley et al. Guess again (and again and again): measuring password strength by simulating password-cracking algorithms. IEEE 2012 Symposium on Security and Privacy.

[12] M. Mazurek et al. Measuring password guessability for an entire university. ACM CCS 2013.

[13] A. Narayanan, V. Schmatikov. Fast dictionary attacks on passwords using time-space tradeoff. ACM CCS 2005.

[14] J.-J. Quisquater, Y.G. Desmedt. Chinese lotto as an exhaustive code-breaking machine. *IEEE Computer* 24(11):14-22, 1991.

[15] S. Schechter, C. Herley, M. Mitzenmacher. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. USENIX HotSec 2010.

[16] M. Weir, S. Aggarwal, B. de Medeiros, B. Glodek. Password cracking using probabilistic context-free grammars. IEEE 2009 Symposium on Security and Privacy.

[17] M. Weir, S. Aggarwal, M. Collins, H. Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. ACM CCS 2010.

[18] Y. Zhang, F. Monrose, M.K. Reiter. The security of modern password expiration: an algorithmic framework and empirical analysis. ACM CCS 2010.