An Administrator's Guide to Internet Password Research*

Dinei Florêncio and Cormac Herley
Microsoft Research, Redmond, USA

Paul C. van Oorschot
Carleton University, Ottawa, Canada

Abstract. The research literature on passwords is rich but little of it directly aids those charged with securing web-facing services or setting policies. With a view to improving this situation we examine questions of implementation choices, policy and administration using a combination of literature survey and first-principles reasoning to identify what works, what does not work, and what remains unknown. Some of our results are surprising. We find that offline attacks, the justification for great demands of user effort, occur in much more limited circumstances than is generally believed (and in only a minority of recently-reported breaches). We find that an enormous gap exists between the effort needed to withstand online and offline attacks, with probable safety occurring when a password can survive 10^6 and 10^{14} guesses respectively. In this gap, eight orders of magnitude wide, there is little return on user effort: exceeding the online threshold but falling short of the offline one represents wasted effort. We find that guessing resistance above the online threshold is also wasted at sites that store passwords in plaintext or reversibly encrypted: there is no attack scenario where the extra effort protects the account.

1 Introduction

Despite the ubiquity of password-protected web sites, research guidance on the subject of running them is slight. Much of the password literature has become specialized, fragmented, or theoretical, and in places confusing and contradictory. Those who administer and set policies can hardly be blamed for being unenthusiastic about publications which document constantly improving attacks on password sites but are largely silent on the question of how they can be defended. Disappointingly little of the accumulating volume of password research directly addresses key everyday issues—what to do to protect web-services, given the realities of finite-resources, imperfect understanding of the threats, and considerable pushback from users.

Do password composition policies work? Does forced password expiration improve security? Do lockouts help protect a service? What do password meters accomplish? The most comprehensive document on these and other questions dates to 1985 [13]. The problem is not that no recent guidance is available; OWASP offers several documents [39, 45, 56]; blogs, trade magazines and industry analyst reports are full of tips, best practices and opinions. Discussions in online fora reliably generate passionate arguments, if little progress. However, much of the available guidance lacks supporting evidence.

We seek to establish what is supported by clear evidence and solid justification. Using a combination of literature survey and ground-up, first-principles reasoning, we identify what is known to work, what is known not to work, and what remains unknown. The end goal is a more useful view of what is known about the implementation, effectiveness, and impacts of choices made in deploying password-related mechanisms for access to services over the web. The target audience is those interested in the intersection of research literature, and the operation, administration and setting of policies for password-protected web-sites.

On the positive side, considerable progress in the last few years has followed from analysis of leaked plaintext datasets. This has provided new evidence challenging many long-held beliefs. Most current password practices reflect historical origins [18]. Some have evolved over time; others should have, but have not. Environments of use, platforms, and user bases have changed immensely. We summarize the literature useful to answer practical questions on the efficacy of policies governing password composition, expiration and account locking.

Some of our findings are surprising. Experts now recognize that traditional measures of strength bear little relation to how passwords withstand guessing, and

*Proceedings of USENIX LISA'14, Nov. 9-14, 2014, Seattle, WA.

can no longer be considered useful; current password policies have not reflected this. We characterize circumstances allowing advanced offline guessing attacks to occur, and find them more limited than is generally realized. We identify an enormous gap between the guessing-resistance needed to withstand online and offline attacks, and note that it is growing. We highlight that strength above that needed to withstand online guessing is effectively wasted at sites that store passwords in plaintext or reversibly encrypted: there is no attack scenario where the extra strength protects the account from an intelligent adversary.

To dispense with a preliminary question: despite long-known shortcomings in both security and usability, passwords are highly unlikely to disappear. The many reasons include the difficulty of finding something better, user familiarity as an authentication front-end (passwords will likely persist as one factor within multi-dimensional frameworks), and the inertia of ubiquitous deployment [9, 30]. Thus the challenges of administering passwords will not fade quietly either, to the disappointment of those hoping that a replacement technology will remove the need to address hard issues.

2 Classifying accounts into categories

A common tactic to allegedly improve the security of password-protected sites is to ask users to expend more effort—choose "stronger" passwords, don't re-use passwords across sites, deploy and administer anti-malware tools, ensure all software on user devices is patched up-to-date, and so on.

If we assume that users have a fixed time-effort budget for "password security" [4], then it is unwise to spend equally on all accounts: some are far more important than others, correspondingly implying greater impact upon account compromise. This motivates determining how to categorize accounts—a subject of surprisingly little focus in the literature. Different categories call for different levels of (password and other) security. Those who decide and administer policies should be aware of what category not only they see their site falling into, but what categories subsets of their users would see it falling into. Note also that some password-protected sites provide no direct security benefit to end-users, e.g., services used to collect data, or which compel email-address usernames to later contact users for marketing-related purposes. Thus views on account and password importance may differ between users and systems administrators or site operators (e.g., see [10]).[1]

Criteria for categorizing accounts: A first attempt to categorize password-based accounts might be based on communication technology—e.g., grouping email passwords as one category. We find this unsuitable to our goals, as some email accounts are throw-aways from a "consequences" point of view, while others are critically important. Accounts may be categorized in many ways, based on different criteria. Our categorization (below) is based largely on potential *consequences* of account compromise, which we expect to be important characteristics in any such categorization:

- (personal) time loss/inconvenience
- (personal) privacy
- (personal) physical security
- (personal/business) financial implications
- (personal/business) reputational damage
- (personal/business) legal implications
- confidentiality of third-party data
- damage to resources (physical or electronic)

The above time loss/inconvenience may result from loss of invested effort or accumulated information, such as contact lists in email or social networking accounts. One lens to view this through is to ask: Would a user invest 10 minutes in trying to recover a lost account, or simply create a new such account from scratch? Another account-differentiating question is: Do users make any effort at all to remember the account password? Note also that the consequences of compromise of an account X may extend to accounts Y and Z (e.g., due to password reuse, email-based password recovery for other accounts, accounts used as single-sign-on gateways).

For use in what follows, and of independent interest, we categorize accounts as follows:

- **don't-care** accounts (*unlocked doors*).
- **low-consequence** accounts (*latched garden doors*).
- **medium-consequence** accounts.
- **high-consequence** accounts (*essential/critical*).
- **ultra-sensitive** accounts (*beyond passwords*).

Details and examples of these categories are given in Table 1; we say little more in this paper about the book-end categories in this spectrum: *don't-care* accounts are suitably named, and *ultra-sensitive* accounts are beyond scope. Within this paper, that leaves us to explore what password policies, user advice, implementation details, and security levels are suitable for accounts in three main

[1] Realistic systems administrators might self-categorize their site, asking: Do users see me as a "bugmenot.com" site? (cf. Table 1)

Category of account	Description	Comments
0: Don't-care	Accounts whose compromise has no impact on users. A compromise of the account at any time would not bother users. Often one-use accounts with trivially weak passwords, or recreated from scratch if needed subsequently. Perhaps the site compels passwords, despite user not seeing any value therein.	The security community and users should recognize that for such accounts, there would be no technical objection to using password `password`, knowing that it provides no security. Such accounts should be isolated from other categories to avoid cross-contamination, e.g., due to password re-use. Users should minimize security-related investments of time and effort—resources are better spent elsewhere. Possible strategies: re-using a single weak password for all such accounts, using distinct passwords written down on one sheet for easy access, and using publicly shared passwords (see: bugmenot.com).
	Generic examples: One-time email accounts (e.g., used for one-off signup, then abandoned). Nuisance accounts for access to "free" news articles or other content.	
1: Low-consequence	Accounts whose compromise has non-severe implications (minimal or easily repaired). Often infrequently used accounts, relatively low-impact if compromised.	Administrators and operators should be realistic in expectations of user commitment to such accounts. Some users may rely almost entirely on a password recovery feature, vs. remembering such account passwords. Users should recognize the place of these between *Don't-care* and *Medium-consequence* accounts.
	Generic examples: Social network accounts (infrequent users). Discussion group accounts (infrequent users). Online newspapers, streaming media accounts (credit card details not stored onsite).	
2: Medium-consequence	Non-trivial consequences but limited, e.g., loss of little-used reputation account, or credit card details stored at online U.S. merchant (direct financial losses limited to $50).	User losses are more in time and effort, than large financial loss or confidentiality breached by document or data loss. User effort directed towards resisting online guessing attacks is well-spent. Unclear if the same holds true re: resisting determined offline guessing attacks. Note: many attack vectors are beyond user control (e.g., browser-based flaws, server compromises).
	Generic examples: Email accounts (secondary). Online shopping sites (credit card details stored onsite). Social network accounts (casual users). Voice or text communication services accounts (e.g., Skype, MSN). Charge-up sites for stored value cards (credit card details stored onsite). Human resources sites giving employees semi-public information.	
3: High-consequence	Critical or essential accounts related to primary employment, finance, or documents requiring high confidentiality. Compromises are not easily repaired or have major consequences/side effects.	Most important password discussion, attention and effort of both sysadmins and users should focus here. Often password protection for such accounts is best augmented by second-factor mechanisms (involving explicit user action) or other dimensions (invisible to user). Stakeholder priorities may differ: an account viewed lower consequence by a user may be categorized essential by their employer (e.g., remote access to a corporate database via a password).
	Generic examples: Email accounts (primary, professional, recovery of other accounts). Major social network/reputational accounts (heavy users and celebrities). Online banking and financial accounts. SSH and VPN passwords for access to corporate networks. Access to corporate databases, including employee personal details.	
∞: Ultra-sensitive	Account compromise may cause major, life-altering, irreversible damage. (Many individual users will have no such accounts.)	It is entirely unsuitable to rely on passwords alone for securing such accounts Passwords if used should be augmented by (possibly multiple) additional mechanisms. The passwords themselves should not be expected to be tangibly different from those for high-consequence accounts (one might argue that weaker passwords suffice, given stronger supplementary authentication mechanisms).
	Generic examples: Multi-million dollar irreversible banking transactions. Authorization to launch military weapons. Encryption of nation-state secrets.	

Table 1: Categories of password-protected accounts, comments and examples. Accounts in the same category ideally have passwords of similar strength relative to guessing attacks. Ultra-sensitive accounts require that passwords be augmented by much more robust mechanisms.

categories of interest: *low-consequence, medium consequence, and high-consequence* accounts.[2]

Use of single term "password" over-loaded? Our discussion of categories highlights that using the unqualified term *password* for protection that runs the gamut from don't-care to high-consequence sites may mislead users. We should not be quick to express outrage on learning that `password1` and `123456` are common on publicly-disclosed password lists from compromised sites, if these are don't-care accounts in users' eyes. Nor should it be surprising to find passwords stored cleartext on fantasy football sites. The use of the same term *password* across all account categories, together with a jumble of unscoped password advice to users, and an absence of discussion of different categories of accounts (and corresponding password requirements), likely contributes to lower overall security, including through cross-category re-use of passwords. We believe finer-grained terminology would better serve users here.

3 Guessing attacks and password storage

The enormous effort that has been spent on password strength and guessing-attacks might lead us to believe that the questions there are largely settled and things are well-understood. Unfortunately, we find that this is not the case. For a number of reasons, realistic studies of password behaviors are hard to conduct [21].

Until recently, published knowledge on in-the-wild password habits [7, 10, 57] was derived from a few small-scale sets of plaintext passwords [38], or studies without access to plaintext passwords [22]. Recent large-scale breaches have provided significant collections of plaintext passwords, allowing study of actual user choices. Tellingly, they reveal that many time-honored assumptions are false. Password "strength" measures both long-used by academics and deeply embedded in criteria used by IT security auditors, are now known to correlate poorly with guessing resistance; policies currently enforced push users toward predictable strategies rather than randomness—e.g., evidence, as discussed below, shows password aging (forced expiration) achieves very little of its hoped-for improvement [63].

Here we explore what is known on measuring password strength, fundamentals of storing passwords, and suitable target thresholds for how many guesses a password should withstand.

Leaked datasets. Table 2 lists several recent leaks from prominent web-sites. These datasets reveal much about user password habits. The ethics of doing analysis on what is, in effect, stolen property generated some discussion when the Rockyou dataset became available in 2009. While none of the 32 million users gave permission for their passwords to be used, rough consensus now appears to be that use of these datasets imposes little or no additional harm, and their use to improve password security is acceptable; the datasets are also of course available to attackers.

Note that among Table 2 incidents, passwords were stored "properly" (see Section 3.4) salted and hashed in just two cases—Evernote and Gawker. Rockyou, Tianya and Cupid Media stored plaintext passwords; LinkedIn and eHarmony stored them hashed but unsalted; Adobe stored them reversibly encrypted. Section 3.2 and Figure 1 explain why offline guessing attacks (beyond rainbow table lookups) are a relevant threat only when the password file is properly salted and hashed.

3.1 Password strength: ideal vs. actual

Security analysis and evaluation would be much simpler if users chose passwords as random collections of characters. For example, if passwords were constrained to be L characters long, and drawn from an alphabet of C characters, then each of C^L passwords would be equally likely. An attacker would have no better strategy than to guess at random and would have probability C^{-L} of being correct on each guess. Even with relatively modest choices for L and C we can reduce the probability of success (per password guess) to 10^{-16} or so—putting it beyond reach of a year's worth of effort at 10 million guess verifications per second.

Unfortunately, the reality is nowhere close to this. Datasets such as the 32 million plaintext Rockyou passwords [64] have revealed that user behavior still forms obvious clusters three decades after attention was first drawn to the problem [38]. Left to themselves users choose common words (e.g., `password`, `monkey`, `princess`), proper nouns (e.g., `julie`, `snoopy`), and predictable sequences (e.g., `abcdefg`, `asdfgh`, `123456`; the latter by about 1% of Rockyou accounts).

This has greatly complicated the task of estimating passwords' resistance to guessing. Simple estimation techniques work well for the ideal case of random collections of characters, but are completely unsuited for user-chosen passwords. For example, a misguided approach models the "entropy" of the password as $\log_2 C^L = L \cdot \log_2 C$ where L is the length, and C is the size of the alphabet from which the characters are drawn (e.g., lowercase only would have $C = 26$, lowercase and digits would have $C = 36$, lower-, uppercase and digits would have $C = 62$ etc). The problems with this approach becomes obvious when we factor in user behavior: `P@ssw0rd` occurs 218 times in the Rock-

[2] An alternate account categorization by Grosse and Upadhyay [28], based on *value*, has categories: throw-away, routine, spokesperson, sensitive and very-high-value transactions.

Site	Year	# Accounts	Hashed	Salted	Reversibly Encrypted	Offline guessing attack beyond rainbow tables needed and possible
Rockyou [64]	2009	32m				N
Gawker	2010	1.3m	✓	✓		Y
Tianya	2011	35m				N
eHarmony	2012	1.5m	✓			N
LinkedIn	2012	6.5m	✓			N
Evernote	2013	50m	✓	✓		Y
Adobe	2013	150m			✓	N
Cupid Media	2013	42m				N

Table 2: Recent incidents of leaked password data files. For only two listed incidents (Evernote and Gawker) would an offline guessing attack be the simplest plausible way to exploit the leak. For each other incident, passwords were stored in such a way that either an easier attack would suffice, or offline guessing was impossible, as explained in Figure 1.

you dataset but has a score of 52.6 under this measure, while `gunpyo` occurs only once and has score 28.2. Thus, a password that is far more common (thus more likely to be guessed) scores much higher than one that is unique—the opposite of what a useful metric would deliver. These are by no means exceptions; a test of how passwords hold up to guessing attacks using the JohntheRipper cracking tool [44] shows that $L \cdot \log_2 C$ correlates very poorly with guessing resistance [32, 60]. Similarly for NIST's crude entropy approximation [13, 60]: many "high-entropy" passwords by its measure turn out to be easily guessed, and many scoring lower withstand attack quite well. Such naive measures dangerously and unreliably estimate how well passwords can resist attack.

The failure of traditional measures to predict guessability has led researchers to alternatives aiming to more closely reflect how well passwords withstand attack. One approach uses a cracking tool to estimate the number of guesses a password will survive. Tools widely used for this purpose include JohntheRipper [44], Hashcat, and its GPU-accelerated sister oclHashcat [42]; others are based on context-free grammars [60, 61]. These tools combine "dictionaries" with *mangling rules* intended to mimic common user strategies: replacing '`s`' with '`$`', '`a`' with '`@`', assume a capital first letter and trailing digit where policy forces uppercase and digits, etc. Thus simple choices like `P@ssw0rd` are guessed very quickly.

Another approach models an optimal attacker with access to the actual password distribution χ (e.g., the Rockyou dataset), making guesses in likelihood order. This motivates *partial guessing metrics* [6] addressing the question: how much work must an optimal attacker do to break a fraction α of user accounts?

Bonneau's α-guesswork gives the expected number of guesses per-account to achieve a success rate of α [7]. Optimal guessing gives dramatic improvements in skewed distributions arising from user-chosen secrets, but none in uniform distributions. For example, for the distribution U_{L6} of random (equi-probable) length-6 lowercase passwords, all can be found in $26^6 \approx 309$ million guesses per account, and 10% in 30.9 million guesses. In contrast, guessing in optimal order on the Rockyou distribution of 32 million passwords, an average of only $7,131$ guesses per account breaks 10% of accounts. Thus, successfully guessing 10% of the Rockyou accounts is a factor of $30,900,000/7131 \approx 4300$ easier than for a length-6 lowercase random distribution (even though the latter is weaker using the $L \cdot \log_2 C$ measure).

Thus, oversimplified "entropy-based" measures should not be relied upon to draw conclusions about guessing resistance; rather, their use should be strongly discouraged. The terms password *strength* and *complexity* are also confusing, encouraging optimization of such inappropriate metrics, or inclusion of certain character classes whether or not they help a password withstand attack. We will use instead the term *guessing resistance* for the attribute that we seek in a password.

3.2 Online and offline guessing

Determining how well a password withstands attack requires some bound on how many guesses the attacker can make and some estimate of the order in which he makes them. The most conservative assumption on guessing order is that the attacker knows the actual password distribution (see α-guesswork above); another approach assumes that he proceeds in the order dictated by an efficient cracker (see also above). We now review how the number of guesses an attacker can make depends on both: (a) the point at which he attacks; and (b) server-side details of how passwords are stored.

The main points to attack a password are: on the client, in the network, at a web-server's public-facing part, and at the server backend. Attacks at the client (e.g., malware, phishing) or in the network (e.g., sniffing) do not generally involve guessing—the password is simply stolen; guess-resistance is irrelevant. Attacks involving guessing are thus at the server's public-face and backend.

Attacks on the server's public-face (generally called *online attacks*) are hard to avoid for a public site—by

design the server responds to authentication requests, checking (username, password) pairs, granting access when they match. An attacker guesses credential pairs and lets the server to do the checking. Anyone with a browser can mount basic online guessing attacks on the publicly facing server—but of course the process is usually automated using scripts and guessing dictionaries.

Attacks on the backend are harder. Recommended practice has backends store not passwords but their salted hashes; recalculating these from user-entered passwords, the backend avoids storing plaintext passwords. (Sections 3.4 and 3.5 discuss password storage details.)

For an offline attack to improve an attacker's lot over guessing online, three conditions must hold.

i) He must gain access to the system (or a backup) to get to the stored password file. Since the backend is designed to respond only one-at-a-time to requests from the public-facing server, this requires evading all backend defences. An attacker able to do this, and export the file of salted-hashes, can test guesses at the rate his hardware supports.

ii) He must go undetected in gaining password file access: if breach detection is timely, then in well-designed systems the administrator should be able to force system-wide password resets, greatly limiting attacker time to guess against the file. (Note: ability to quickly reset passwords requires nontrivial planning and resources which are beyond scope to discuss.)

iii) The file must be properly both salted and hashed. Otherwise, an offline attack is either not the best attack, or is not possible (as we explain next).

If the password file is accessed, and the access goes undetected, then four main possibilities exist in common practice (see Figure 1):

1) *the file is plaintext*. In this case, offline guessing is clearly unnecessary: the attacker simply reads all passwords, with nothing to guess [64].

2) *the file is hashed but unsalted*. Here the passwords cannot be directly read, but *rainbow tables* (see below) allow fast hash reversal for passwords within a fixed set for which a one-time pre-computation was done. For example, 90% of LinkedIn's (hashed but unsalted) passwords were guessed in six days [26].

3) *the file is both salted and hashed*. Here an offline attack is both possible and necessary. For each guess against a user account the attacker must compute the salted hash and compare to the stored value. The fate of each password now depends on how many guesses it will withstand.

Position	Password	Per-guess success probability
10^0	`123456`	9.1×10^{-3}
10^1	`abc123`	5.2×10^{-4}
10^2	`princesa`	1.9×10^{-4}
10^3	`cassandra`	4.2×10^{-5}
10^4	`sandara`	5.3×10^{-6}
10^5	`yahoo.co`	7.2×10^{-7}
10^6	`musica17`	9.4×10^{-8}
10^7	`tilynn06`	3.1×10^{-8}

Table 3: Passwords from the Rockyou dataset in position 10^m for $m = 0, 1, 2, \cdots, 7$. Observe that an online attacker who guesses in optimal order sees his per-guess success rate fall by five orders of magnitude if he persists to 10^6 guesses.

4) *the file has been reversibly encrypted*. This case has two paths: the attacker either gets the decryption key (Case 4A), or he does not (Case 4B).

In Case 4A, offline attack is again unneeded: decryption provides all passwords. In Case 4B, there is no effective offline attack: even if the password is `123456`, the attacker has no way of verifying this from the encrypted file without the key (we assume that encryption uses a suitable algorithm, randomly chosen key of sufficient length, and standard cryptographic techniques, e.g., initialization, modes of operation, or random padding, to ensure different instances of the same plaintext password do not produce identical ciphertext [36]). Even having stolen the password file and exported it without detection, the attacker's best option remains online guessing at the public-facing server; properly encrypted data should not be reversible, even for underlying plaintext (passwords here) that is far from random.

In summary, Figure 1 shows that offline guessing is a primary concern only in the narrow circumstance when all of the following apply: a leak occurs, goes undetected,[3] and the passwords are suitably hashed and salted (cf. [8, 64]). In all other common cases, offline attack is either impossible (guessing at the public-facing server is better) or unneeded (the attacker gets passwords directly, with no guessing needed).

Revisiting Table 2 in light of this breakdown, note that of the breaches listed, Evernote and Gawker were the only examples where an offline guessing attack was necessary; in all other cases a simpler attack sufficed, and thus guessing resistance (above that necessary to resist online attack) was largely irrelevant due to how passwords were stored.

Rainbow tables. To understand the importance of salting as well as hashing stored passwords, consider the attacker wishing to reverse a given hashed password. Starting with a list (dictionary) of N candidate passwords, pre-computing the hash of each, and stor-

[3] Or similarly, password reset capability is absent or unexercised.

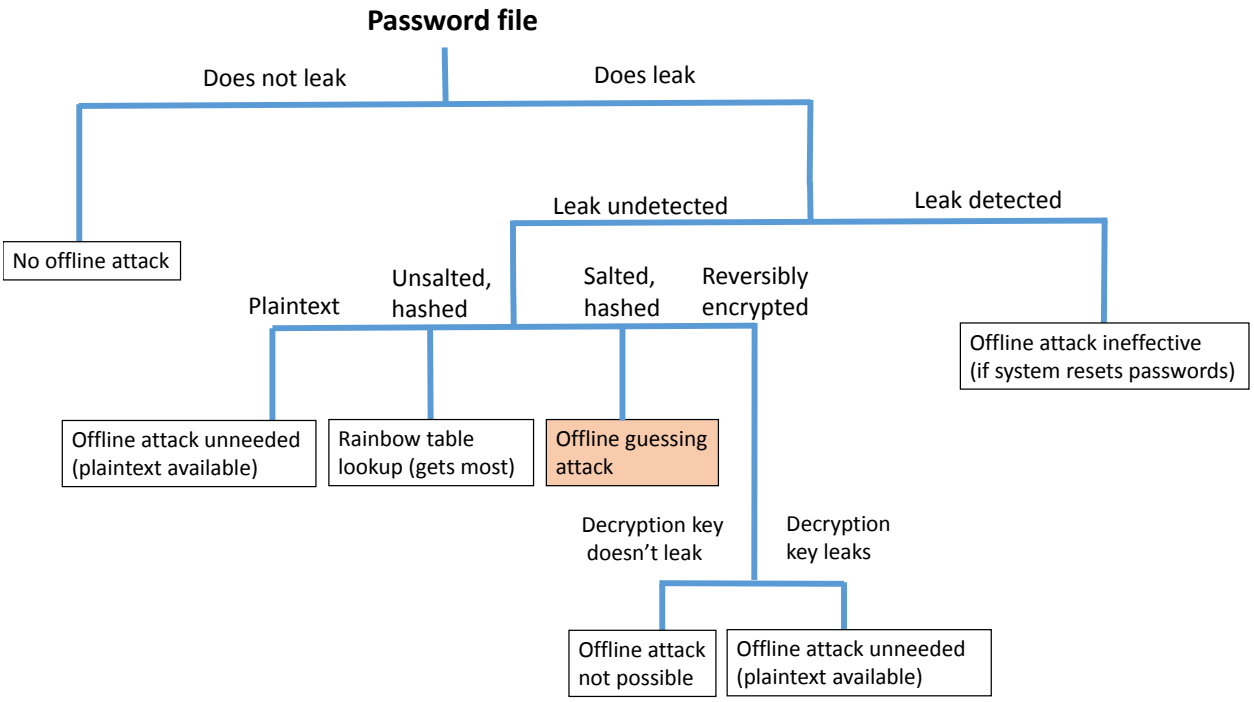

Figure 1: Decision tree for guessing-related threats in common practice based on password file details. Offline guessing is a threat when the password file leaks, that fact goes undetected, and the passwords have been properly salted and hashed. In other cases, offline guessing is either unnecessary, not possible, or addressable by resetting system passwords. Absent a hardware security module (HSM), we expect that the "Decryption key doesn't leak" branch is rarely populated; failure to prevent theft of a password file gives little confidence in ability to protect the decryption key.

Length	Character set	Full cardinality
12	lower	$26^{12} = 2^{56.4}$
10	lower, upper	$52^{10} = 2^{57.0}$
9	any	$95^{9} = 2^{59.1}$
10	lower, upper, digit	$62^{10} = 2^{59.5}$

Table 4: Number of elements targeted by various rainbow tables.

ing each pair sorted by hash, allows per-instance reversal by simple table lookup after one-time order-N pre-computation—and order-N storage. To reduce storage, *rainbow tables* [43] use a series of functions to pre-compute repeatable sequences of password hashes called *chains*, storing only each chain's first and last plaintext. Per-instance computation later identifies hashes from the original fixed password list to a chain, allowing reversal in greater, but still reasonable, time than had all hashes been stored. Numerous rainbow table implementations and services are available.[4] For rainbow tables targeting selected passwords compositions, Table 4 lists as reference points the targeted number of passwords, which give a lower bound on pre-computation time (resolving expected "collisions" increases computation time).

Modifications [40] may allow tables for any efficiently enumerable password space, e.g., based on regular expressions for defined patterns of lower, upper, digits and special characters; this would extend attacks from (naive) brute-force spaces to "smart dictionaries" of similar size but containing higher-likelihood user passwords. We emphasize that *offline attacks using pre-computations over fixed dictionaries, including rainbow tables, are defeated by proper salting, and require leaked password hashes.*

[4]For example, see `http://project-rainbowcrack.com` or `sourceforge.net/projects/ophcrack` among others.

3.3 How many guesses must a password withstand?

Recall that the online attacker's guesses are checked by the backend server, while an offline attacker tests guesses on hardware that he controls. This constrains online attacks to far fewer guesses than is possible offline.

Online guessing (breadth-first). Consider the online attacker. For concreteness, assume a guessing campaign over a four-month period, sending a guess every 1s at a sustained rate, yielding about 10^7 guesses; we use this as a very loose upper bound on the number of online guesses any password might have to withstand. An attacker sending guesses at this rate against all accounts (a breadth-first attack) would likely simply overwhelm servers: e.g., it is unlikely that Facebook's servers could handle simultaneous authentication requests from all users. (In practice, but a tiny fraction authenticate

in any 1s period.) Second, if we assume that the average user attempts authentication k times/day and fails 5% of the time (due to typos, cached credentials after a password change, etc.) then a single attacker sending one guess per-account per-second would send $86,000/k$ times more traffic and $1.73 \times 10^7/k$ more fail events than the entire legitimate user population combined. Even if $k = 100$ (e.g., automated clients re-authenticating every 15 minutes) our single attacker would be sending a factor of 860 more requests and 1.73×10^5 more fails than the whole legitimate population. Malicious traffic at this volume against any server is hard to hide. A more realistic average of $k = 1$ makes the imbalance between malicious and benign traffic even more extreme. Thus 10^7 guesses per account seems entirely infeasible in a breadth-first online guessing campaign; 10^4 is more realistic.

Online guessing (depth-first). What about depth-first guessing—is 10^7 guesses against a single targeted account feasible? First, note that most individual accounts are not worthy of targeted effort. Using the Section 2 categories, low- and medium-consequence sites may have very few such accounts, while at high-consequence sites a majority might be worthy. Second, 10^7 guesses would imply neither a lockout policy (see Section 4.4) nor anything limiting the rate at which the server accepts login requests for an account. Third, as evident from Table 3 which tabulates the passwords in position 10^m from the Rockyou distribution for $m = 0, 1, 2, \cdots, 7$, an online attacker making guesses in optimal order and persisting to 10^6 guesses will experience five orders of magnitude reduction from his initial success rate. Finally, IP address blacklisting strategies may make sending 10^7 guesses to a single account infeasible (albeit public IP addresses fronting large numbers of client devices complicate this). Thus, the assumptions that would allow 10^7 online guesses against a single account are extreme—effectively requiring an absence of defensive effort. 10^6 seems a more realistic upper bound on how many online guesses a password must withstand in a depth-first attack (e.g., over 4 months). This view is corroborated by a 2010 study of password policies, which found that Amazon.com, Facebook, and Fidelity Investments (among many others) allow 6-digit PIN's for authentication [23]. That these sites allow passwords which will not (in expectation) survive 10^6 guesses suggests that passwords that will survive this many guesses can be protected from online attacks (possibly aided by backend defenses). Figure 2 depicts our view: we gauge the online guessing risk to a password that will withstand only 10^2 guesses as extreme, one that will withstand 10^3 guesses as moderate, and one that will withstand 10^6 guesses as negligible. The left curve does not change as hardware improves.

Offline guessing. Now consider the offline attacker: using hardware under his control, he can test guesses at a rate far exceeding online attacks. Improvements in processing power over time make it possible that his new hardware computes guesses orders of magnitude faster than, say, 10-year-old authentication servers which process (online) login attempts. The task is also distributable, and can be done using a botnet or stolen cloud computing resources. An attacker might use thousands of machines each computing hashes thousands of times faster than a target site's backend server. Using a GPU able to compute 10 billion raw hashes/s or more [18, 26], a 4-month effort yields 10^{17} guesses; 1,000 such machines allows 10^{14} guesses on each of a million accounts, or 10^{20} on a single account—all assuming no defensive iterated hashing, which Section 3.4 explores as a means to reduce such enormous offline guess numbers.

Given the lack of constraints, it is harder to bound the number of guesses, but it is safe to say that offline attacks can test many orders of magnitude more guesses than online attacks. Weir et al. [60] pursue cracking up to 10^{11} guesses; a series of papers from CMU researchers investigate as far as 10^{14} guesses [32, 35]. To be safe from offline guessing, we must assume a lower bound of at least 10^{14}, and more as hardware[5] and cracking methods improve. This is illustrated in Figure 2.

Online-offline gap. To summarize, a huge chasm separates online and offline guessing. *Either* an attacker sends guesses to a publicly-facing server (online) *or* guesses on hardware he controls (offline)—there is no continuum of possibilities in between. The number of guesses that a password must withstand to expect to survive each attack differs enormously. A threshold of at most 10^6 guesses suffices for high probability of surviving online attacks, whereas at least 10^{14} seems necessary for any confidence against a determined, well-resourced offline attack (though due to the uncertainty about the attacker's resources, the offline threshold is harder to estimate). These thresholds for probable safety differ by 8 orders of magnitude. The gap increases if the offline attack brings more distributed machines to bear, and as offline attacks and hardware improve; it decreases with hash iteration. Figure 2 conceptualizes the situation and Table 5 summarizes.

Next, consider the incremental benefit received in improving a password as a function of the number of guesses it can withstand (10^m). Improvement delivers enormous gain when $m \leq 3$: the risk of online attack is falling sharply in this region, and safety (from online guessing) can be reached at about $m = 6$. By the time $m = 6$, this effect is gone; the risk of online attack is now minimal, but further password improvement buys little protection against offline attacks until $m = 14$ (where

[5]Hardware advances can be partially counteracted by increased hash iteration counts per Section 3.4.

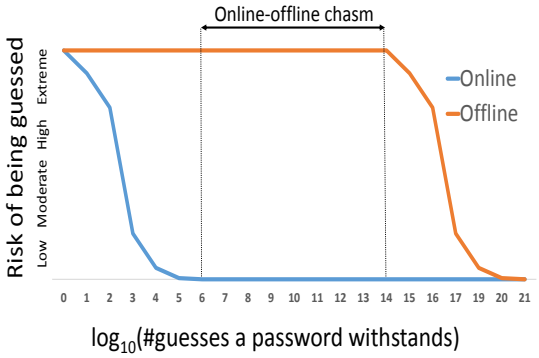

Figure 2: Conceptualized risk from online and offline guessing as a function of the number of guesses a password will withstand over a 4-month campaign. In the region from 10^6 to about 10^{14}, improved guessing-resistance has little effect on outcome (online or offline).

the probability of offline guessing success starts to decline). Note the large gap or *chasm* where online guessing is a negligible threat but surviving offline guessing is still far off. *In this gap, incrementally increasing the number of guesses the password will survive delivers little or no security benefit.*

For example, consider two passwords which withstand 10^6 and 10^{12} guesses respectively. Unless we assume the offline attacker lacks motivation or resources (and gives up early), there is no apparent scenario in which the extra guess-resistance of the second password helps. For example, a password like `tincan24` (which will survive more than a million guesses derived from the Rockyou distribution) and one like `7Qr&2M` (which lives in a space that can be exhausted in $(26+26+10+30)^6 = 92^6 < 10^{12}$ guesses) fare the same: both will survive online guessing, but neither will survive offline attack. Equally, a 7-digit and a 13-digit random PIN have similar security properties for the same reason. If we assume additional guess-resistance comes at the cost of user effort [4, 25, 29], then the effort in the second case appears entirely wasted. In this case, *an effort-conserving approach is to aim to withstand online attacks, but not put in the extra effort to withstand offline attacks.* In fact there is evidence that many sites abandon the idea of relying on user effort as the defence against offline attacks; i.e., they appear to make little effort to force users to reach the higher threshold [23]. Sections 4.1 and 4.2 consider the efficacy of password composition policies and blacklists.

Frequency of online vs. offline attacks. Authoritative statistics on the relative frequency of online attacks compared to offline attacks do not exist. However it is clear that online attacks can be mounted immediately against public web sites (such attacks are more efficient when known-valid userids are obtained a priori [24]), while offline attacks require that the password hashes be available to the attacker (e.g., leaked file).

3.4 Storage and stretching of passwords

As Section 3.2 stated, storing salted hashes [48] of passwords beats most alternatives. In all other common storage options, advanced offline guessing attacks are either unnecessary (simpler attacks prevail) or impossible. Use of site-wide (global) salt defeats generic rainbow tables; per-account salts (even userid) and iteration counts, storable with hashes, provide further protection. Unfortunately, for various reasons, hashing is far from universal (e.g., perhaps 40% of sites do not hash [10]).

Guessing is resource-intensive—an offline attack may involve billions of guesses per-account, whereas a webserver verifies login attempts only on-demand as users seek to log in. Since early UNIX, this has been leveraged defensively by hash functions designed to be slow, by iteration: repeatedly computing the hash of the hash of the salted password. Such *key stretching* was formally studied by Kelsey et al. [33]; we reserve the term *key strengthening* for the idea, with related effect, of using a random suffix salt that verifiers must brute-force. Iterating 10^n times slows offline attack by n orders of magnitude; this configurable factor should be engineered to add negligible delay to users, while greatly increasing an offline attacker's work. Factors of 4,000–16,000 iterations already appear in practice. Our estimates for the number of guesses an offline attacker can send assumed no iteration; hash iteration narrows the online-offline chasm.

Salting also removes one form of "parallel attack": if two users have the same password, this will not be apparent and cannot be exploited to simplify attacks (assuming proper salting and hashing, e.g., salts of sufficient length, and passwords not truncated before hashing).

Practical instantiations [18] of key stretching (via so-called *adaptive key derivation functions*) include `bcrypt` [48], supported first in OpenBSD with 128-bit salt and configurable iteration count to acceptably adjust delay and server load on given platforms, and allow for hardware processing advances; the widely used standard `PBKDF2` (part of PKCS #5 v2.0 and RFC 2898); and the newer `scrypt` [46] designed to protect against custom-hardware attacks (e.g., ASIC, FPGA, GPU).

Keyed hashing. Reversible encryption is one of the worst options for storing passwords if the decryption key leaks, but is among the best if a site can guarantee that it never leaks (even if the password file itself does). Justification for sites to store passwords reversibly encrypted is a need to support legacy protocols (see Section 3.5). Absent such legacy requirements, the best solution is salting and iterated hashing with a message authentication code (MAC) [37, 56] stored instead of a hash; password verification (and testing of guesses) is then impossible

without crypto key access. The difficulty of managing keys should not be understated—too often keys stored in software or a configuration file are found by attackers, explaining the common use of a one-way hash over reversible encryption. However, if the MAC of a salted, iterated password hash is all that is stored, then even if the MAC key leaks, security is equal to a salted iterated hash; and that risk falls away if a hardware security module (HSM) is used for MAC generation and verification.

3.5 Availability of passwords at the server

So salted hashes are a preferred means to store passwords, and (cf. Figure 1) an attacker who has access to the password file, and exports it undetected, still faces a computationally expensive offline attack. A site suffering this severe, undetected breach fares far better than one with plaintext or hashed-unsalted passwords, or reversibly encrypted passwords and a leaked decryption key. Nonetheless, many sites use a non-preferred means of storing passwords, e.g., there is a "Hall of Shame" of sites[6] which mail forgotten passwords back to users and thus store them either plaintext or reversibly encrypted. While the practice is inadvisable for high-consequence sites, as Section 2 notes, one size clearly does not fit all.

In addition to sites which mail-back passwords, recent breaches clearly signal that storing plaintext passwords is not uncommon. In Table 2's list of recent server leaks, only two used salted hashes. Failure to store passwords as salted hashes may be due to confusion, failure to understand the advantages, or a conscious decision or software default related to legacy applications or protocols as we explain next.

RADIUS (Remote Authentication Dial In User Service) is a networking protocol widely used to provide dial-in access to corporate and university networks. Early protocols that allowed client machines to authenticate such as Password Authentication Protocol (PAP) and Challenge-Handshake Authentication Protocol (CHAP) over RADIUS require passwords be available (to the server) in-the-clear or reversibly encrypted. Thus, sites that supported such clients must store passwords plaintext or reversibly encrypted. Support for protocols that supercede PAP and CHAP in commodity OS's began only circa 2000. Thus, many sites may have had to support such clients at least until a decade or so later.

Universities provide interesting examples. Recent papers by groups researching passwords make clear that several universities were, at least until recently, storing passwords reversibly encrypted or as unsalted hashes. Mazurek et al. (CMU) state [35]: "The university was using a legacy credential management system (since abandoned), which, to meet certain functional requirements, reversibly encrypted user passwords, rather than using salted, hashed records." Fahl et al. (Leibniz University) state [21]: "The IDM system stored up to five unique passwords per user using asymmetric cryptography, so it would be possible to decrypt the passwords to do a security analysis." Zhang et al. (UNC) state [63]: "The dataset we acquired contains 51,141 unsalted MD5 password hashes from 10,374 defunct ONYENs (used between 2004 and 2009), with 4 to 15 password hashes per ONYEN, i.e., the hashes of the passwords chosen for that ONYEN sequentially in time."[7]

Figure 1 makes clear that if the password is to be available at the backend (i.e., stored plaintext or reversibly encrypted) then an offline attack is either unnecessary or impossible. Thus, any resistance to guessing above and beyond that needed to withstand online attacks is wasted (in no scenario does the extra guessing resistance protect the account from competent attackers). Thus sites that impose restrictive password policies on their users while storing passwords plaintext or reversibly encrypted are squandering effort. An example appears to be a documented CMU policy [35]: passwords had to be greater than length 8 and include lower, upper, special characters and digits. This policy appears designed to withstand an offline guessing attack which (since passwords were reversibly encrypted) had no possibility of occurring, and thus imposes usability cost without security benefit.

We do not know how common it is for sites to store passwords plaintext or reversibly encrypted. Large breaches, such as in Table 2, continue to make clear that plaintext is common among low- and medium-consequence sites. The data from CMU and Leibniz hint that far from being rare exceptions, reversible encryption of passwords may also be quite common. If true, this would imply that many sites with strict composition policies are engaged in a large-scale waste of user effort based on confused thinking about guessing resistance.

3.6 Other means to address offline attacks

Online guessing attacks seem an unavoidable reality for Internet assets protected by passwords, while offline attacks occur only in a limited set of circumstances. The guessing resistance needed to withstand these two types of attacks differs enormously (recall Section 3.3). Significant effort has been devoted to getting users to choose better passwords. If an online attacker can send at most 10^6 guesses per account, then it is relatively easy (e.g., password blacklists) to resist online guessing. Thus, getting users to choose passwords that will withstand over 10^6 guesses is an effort to withstand offline attacks, not online.

[6] See `http://plaintextoffenders.com`.

[7] An "ONYEN" is a userid ("Only Name You'll Ever Need") in the single-sign-on system studied.

There are ways to address offline attacks that do not involve persuading users to choose better passwords. Figure 1 makes clear that if the file doesn't leak, or the leak is detected and existing passwords are immediately disabled, things are very different. Thus alternate approaches include those that protect the password file, or allow detection of leaks—neither requiring changes in user behaviour.

Crescenzco et al. [15] give a method to preclude an offline attack, even if an attacker gains unrestricted access to the backend server. It hinges on the fact that an offline attacker must guess at a rate far exceeding the normal authentication requests from the user population (cf. Section 3.3). They introduce a novel hashing algorithm that requires randomly indexing into a large collection of random bits (e.g., 1 TByte). Ensuring that the only physical connection to the server with the random bits is matched to the expected rate of authentication requests from the user population guarantees that the information needed to compute the hashes can never be stolen. While the scheme is not standard, it illustrates that ingenious approaches to prevent password file leaks are possible (thereby eliminating the possibility of offline attacks).

Leaked password files can also be detected by spiking password files with *honeywords*—false passwords which are salted, hashed and indistinguishable from actual user passwords [31]. An offline attack which attempts authentication with a "successfully" guessed honeyword alerts administrators of a breached password file, signalling that in-place recovery plans should commence.

4 Password policies and system defences

4.1 Composition and length policies

Many approaches have been tried to force users to choose better passwords. The most common are policies with length and character-group composition requirements. Many sites require passwords of length at least 8, with at least three of four character types (lower- and uppercase, digits, special characters) so that each password meets a lower bound by the measure $L \cdot \log_2 C$. However, as Section 3.1 explains, this naive entropy-motivated metric very poorly models password guessing-resistance [32, 60]. Users respond to composition policies with minimally compliant choices such as `Pa$$w0rd` and `Snoopy2`. Passwords scoring better by this metric are not guaranteed to fare better under guessing attacks. On examining this, Weir et al. [60] conclude "the entropy value doesn't tell the defender any useful information about how secure their password creation policy is."

Recent gains in understanding guess-resistance come largely from analysis of leaked datasets [7, 60]. Since it appears (including by examining the actual cleartext passwords) that none of Table 2's listed sites imposed strict composition policies on users, we cannot directly compare collections of passwords created with and without composition polices to see if the policy has a significant effect. However, Weir et al. [60] compare how subsets of the Rockyou dataset that comply with different composition policies fare on guessing resistance. (The exercise is instructive, but we must beware that a subset of passwords that comply with a policy are not necessarily representative of passwords created under that policy; cf. [32].) They found that passwords containing an uppercase character are little better at withstanding guessing than unrestricted passwords: 89% of the alpha strings containing uppercase were either all uppercase, or simply had the first character capitalized (cf. [35]). They conclude that forcing an uppercase character merely doubles the number of guesses an intelligent attacker would need. Fully, 14% of passwords with uppercase characters did not survive 50,000 guesses—thus providing inadequate protection even against online attackers.

Including special characters helped more: of passwords incorporating one, the number that did not survive 50,000 guesses dropped to 7%. But common patterns revealed by their analysis (e.g., 28.5% had a single special character at the end) were not fully exploited by the guessing algorithm, so this survival rate is optimistic. Thus including special characters likewise does not protect robustly even against online attacks.

Kelley et al. [32] examine passwords created by 12,000 participants in a Mechanical Turk study under 8 different composition policies including: basic-length-8, basic-length-16, and length-8 mandating all of lower, upper, digits and special characters. They use a variety of cracking algorithms to evaluate guessing resistance of various passwords. Interestingly, while there is enormous variation between the fate of passwords created under different policies at high guess numbers (e.g., 58% of basic-length-8, but only 13% of basic-length-16 passwords were found after 10^{13} guesses) there was less variation for numbers of guesses below 10^6. Also, in each of the policies tested, fewer than 10% of passwords fell within the first 10^6 guesses (our online threshold).

Mazurek et al. [35] examine 25,000 passwords (from a university single sign-on system) created under a policy requiring at least length-8 and mandating inclusion of lower, upper, special characters and digits, and checks against a dictionary. The cracking algorithms tested achieved minimal success until 10^7 guesses, but succeeded against about 48% of accounts by 10^{14} guesses. Depending as they do on a single cracking algorithm, these must be considered the worst-case success rates for an attacker; it is quite possible that better tuning would greatly improve attack performance. In particular, it is not safe to assume that this policy ensures good survival

Attack		Guesses	Recommended defenses
Online guessing	Breadth-first	10^4	Password blacklist; rate-limiting; account lock-out; recognition of known devices (e.g., by browser cookies, IP address recognition)
	Depth-first	10^6	
Offline guessing	Breadth-first	10^{14}	Iterated hashing; prevent leak of hashed-password file; keyed hash functions with Hardware Security Module support (Sections 3.4, 3.6)
	Depth-first	10^{20}	
Rainbow table lookup (using extensive pre-computation)		n/a	Salting; prevent leak of hashed-password file
Non-guessing (phishing, keylogging, network sniffing)		n/a	Beyond scope of this paper

Table 5: Selected attack types, number of per-account guesses expected in moderate attacks, and recommended defenses. We assume a 4-month guessing campaign, and for offline guessing that the password file is salted and hashed (see Section 3.4). Rate-limiting includes delays and various other techniques limiting login attempts over fixed time periods (see Section 4.4). Rainbow tables are explained in Section 3.2.

up to 10^7 guesses, since most cracking algorithms optimize performance at high rather than low guess numbers.

In answering whether password policies work, we must first decide what it is we want of them. We use Section 3.3 which argued that safety from depth-first online guessing requires withstanding 10^6 guesses, while safety from offline guessing requires 10^{14} or more. There are many tools that increase resistance to online guessing; some offer a simple way to protect against online guessing with lower usability impact than composition policies—e.g., password blacklists (see Section 4.2).

The above and further evidence suggest that composition policies are mediocre at protecting against offline guessing. For example, over 20% of CMU passwords were found in fewer than 10^{11} guesses, and 48% after 10^{14} [35]. While the stringent policy (minimum length eight and inclusion of all four character classes) has forced half of the population to cross the online-offline chasm, for practical purposes this is still failure: we expect most administrators would regard a site where half of the passwords are in an attacker's hands as being 100% compromised. Of policies studied by Kelley et al. [32] only one that required 16 characters gave over 80% survival rate at 10^{14} guesses.

Thus, the ubiquity of composition policies (which we expect stems from historical use, zero direct system cost, and the ease of giving advice) is at odds with a relatively modest delivery: they help protect against online attacks, but alternatives seem better. Some policies increase guess-resistance more than others, but none delivers robust resistance against the level of guessing modern offline attacks can bring to bear. Given that no aspect of password security seems to incite comparable user animosity [1, 14, 41], and that this is exacerbated by the rise of mobile devices with soft keyboards, composition policies appear to offer very poor return on user effort.

4.2 Blacklists and proactive checking

Another method to avoid weak passwords is to use a blacklist of known bad choices which are forbidden, sometimes called *proactive password checking* [5, 55]. This can be complementary or an alternative to a composition policy. Microsoft banned common choices for hotmail in 2011. In 2009, Twitter banned a list of 370 passwords, which account for (case insensitive) 5.2% of Rockyou accounts; simply blocking these popular passwords helps a significant fraction of users who would otherwise be at extreme risk of online guessing.

Also examining efficacy, using a blacklist of 50,000 words Weir et al. [60] found that over 99% of passwords withstood 4,000 guesses; 94% withstood 50,000. Thus, a simple blacklist apparently offers excellent protection against breadth-first online attacks and good improvement for depth-first online attacks.

Blacklists of a few thousand, even one million passwords, can be built by taking the commonest choices from leaked distributions. At 10^6 they may offer excellent protection against all online attacks. However, they do not offer much protection against offline attacks. Blacklists of size 10^{14} appear impractical. A significant annoyance issue also increases with list size [35]: users may understand if a few thousand or even 10^6 of the most common choices are forbidden, but a list of 10^{14} appears capricious and (in contrast to composition policies) it is not possible to give clear instructions on how to comply.

As an advantage of blacklists, they inconvenience only those most at risk. 100% of users using one of Twitter's 370 black-words is highly vulnerable to online guessing. By contrast, forcing compliance with a composition policy inconveniences all users (including those with long lowercase passwords that resist offline guessing quite well [35]) and apparently delivers little.

There is a risk that a static blacklist lacks currency; band names and song lyrics can cause popularity surges that go unrepresented—e.g., the 16 times that `justinbieber` appears in the 2009 Rockyou dataset would likely be higher in 2014. Also, even if the top 10^6 passwords are banned, something else becomes the new most common password. The assumption is that banning the current most popular choices results in a distribution that is less skewed; this assumption does not seem strong, but has not been empirically verified. In one proposed password storage scheme that limits the popularity of any password [54], no more than T users of a site are allowed to have the same password (for a configurable

threshold T); this eliminates the currency problem and reduces the head-end password distribution skew.

4.3 Expiration policies (password aging)

Forced password change at regular intervals is another well-known recommendation, endorsed by NIST [13] and relatively common among enterprises and universities, albeit rarer among general web-sites. Of 75 sites examined in one study [23], 10 of 23 universities forced such a policy, while 4 of 10 government sites, 0 of 10 banks, and 0 of 27 general purpose sites did so.

The original justification for password aging was apparently to reduce the time an attacker had to guess a password. Expiration also limits the time that an attacker has to exploit an account. Ancillary benefits might be that it forces users into a different password selection strategy, e.g., if passwords expire every 90 days, it is less likely that users choose very popular problematic choices like `password` and `abcdefg` and more likely that they develop strategies for passwords that are more complex but which can be modified easily (e.g., incrementing a numeric substring). As a further potential benefit, it makes re-use between accounts less likely—whereas reusing a static password across accounts is easy and common [22], forced expiration imposes co-ordination overhead for passwords re-used across sites.

Reducing guessing time is relevant for offline attacks (an online guesser, as noted, gets far fewer attempts). So any benefit against guessing attacks is limited to cases where offline guessing is a factor, which Section 3.2 argues are far less common.

Reducing the time an attacker has to exploit an account is useful only if the original avenue of exploitation is closed, and no alternate (backdoor) access means has been installed. When the NIST guidelines were written, guessing was a principal means of getting a password. An attacker who had successfully guessed a password would be locked out by a password change; he would have to start guessing anew. Several factors suggest that this benefit is now diminished. First, offline password guessing is now only one avenue of attack; if the password is gained by keylogging-malware, a password change has little effect if the malware remains in place. Second, even if the attack is offline guessing, expiration turns out to be less effective than believed. Zhang et al. [63] recently found many new passwords very closely related to old after a forced reset; they were given access to expired passwords at UNC and allowed (under carefully controlled circumstances) to submit guesses for the new passwords. The results are startling: they guessed 17% of passwords in 5 tries or fewer, and 41% of accounts in under 3 seconds of offline attacking.

Thus, with forced expiration, new passwords appear to be highly predictable from old, and the gain is slight, for a policy competing with composition rules as most-hated by users. The benefits of forcing users to different strategies of choosing passwords, and making re-use harder may be more important. Given the severe usability burden, and associated support costs, expiration should probably be considered only for the top end of the high-consequence category.

4.4 Rate-limiting and lockout policies

A well-known approach to limiting the number of online attack guesses is to impose some kind of lockout policy—e.g., locking an account after three failed login attempts (or 10, for a more user-friendly tradeoff [12]). It might be locked for a certain period of time, or until the user takes an unlocking action (e.g., by phoning, or answering challenge questions). Locking for an hour after three failed attempts reduces the number of guesses an online attacker can make in a 4-month campaign to $3 \times 24 \times 365/3 = 8,760$ (cf. Section 3.3). A related approach increasingly delays the system response after a small number of failed logins—to 1s, 2s, 4s and so on. Bonneau and Preibusch [10] found that in practice, very few sites block logins even after 100 failed logins (though the sites they studied were predominantly in the low and medium consequence categories). Secret questions (Section 4.6), if used, must similarly be throttled.

The two main problems with lockout policies are the resulting usability burden, and the denial of service vulnerability created. Usability is clearly an issue given that users forget passwords a great deal. The denial of service vulnerability is that a fixed lockout policy allows an attacker to lock selected users out of the site. Incentives may mean that this represents a greater problem for some categories of sites than others. An online auction user might lockout a rival as a deadline approaches; someone interested in mayhem might lock all users of an online brokerage out during trading hours.

Throttling online guessing while avoiding intentional service lockouts, was explored by Pinkas and Sander [47] and extended by others [2, 59]. Login attempts can be restricted to devices a server has previously associated with successful logins for a given username, e.g., by browser cookies or IP address; login attempts from other devices (assumed to be potential online guessing machines) require both a password and a correctly-answered CAPTCHA. Through a clever protocol, legitimate users logging in from new devices see only a tiny fraction of CAPTCHAs (e.g., 5% of logins from a first-time device). The burden on online guessers is much larger, due to a vastly larger number of login attempts. The downside of this approach is CAPTCHA usability.

4.5 Password meter effectiveness

In addition to offering tips or advice on creating good passwords, many large sites employ password meters, purportedly measuring password strength, in an attempt to nudge users toward better passwords. They are generally implemented in Javascript in the browser, severely limiting the complexity of the strength-estimation algorithm implemented—e.g., downloading a very large dictionary to check against is problematic. Thus many meters use flawed measures (see Section 3.1) which correlate poorly with guessing resistance. This also produces many incongruities, e.g., classifying `Pa$$w0rd` as "very strong" and `gunpyo` as "weak". Of course, deficiencies in currently deployed meters do not necessarily imply that the general idea is flawed.

Among recent studies of the efficacy of meters, Ur et al. [58] examined the effect of various meters on 2,931 Mechanical Turk users, finding that significant increases in guessing-resistance were only achieved by very stringent meters. The presence of any meter did however provide some improvement even in resistance to online attacks (i.e., below 10^6 guesses). De Carnavelet and Mannan [17] compare several password meters in common use and find enormous inconsistencies: passwords being classified as strong by one are termed weak by another. Egelman et al. [20] explore whether telling users how their password fares relative to others might have a greater effect than giving an absolute measure. Those who saw a meter tended to choose stronger passwords than those who didn't, but the type of meter did not make a significant difference. In a post-test survey 64% of participants admitted reusing a password from elsewhere—such users may have been influenced to re-use a different old password, but every old password is obviously beyond the reach of subsequent influences.

4.6 Backup questions & reset mechanisms

Reset mechanisms are essential at almost every password-protected site to handle forgotten passwords. For most cases, it can be assumed the user still has access to a secondary communication channel (e.g., an e-mail account or phone number on record)—and the assumed security of that channel can be leveraged to provide the reset mechanism. A common practice is to e-mail back to the user either a reset link or temporary password.

Sites that store passwords cleartext or reversibly encrypted can e-mail back that password itself if forgotten, but this exposes the password to third parties. Mannan et al. [34] propose to allow forgotten passwords to be restored securely; the server stores an encrypted copy of the password, with the decryption key known to a user recovery device (e.g., smartphone) but not the server.

Many sites use backup authentication questions (*secret questions*) instead of, or in conjunction with, emailing a reset link. The advantage of doing both is that an attacker gaining access to a user's e-mail account could gain access to any sites that e-mail reset links. Different categories of accounts (see Section 2) must approach this question differently. For high-consequence accounts, it seems that backup questions should be asked to further authenticate the user; for lower consequence accounts, the effort of setting up and typing backup questions must be taken into account.

When a secondary communication channel is unavailable (e.g., the site in question is the webmail provider itself, or a secondary communication channel was never set up, or is no longer available) backup questions are widely used. Unfortunately, plentiful evidence [49, 53] shows that typically in practice, the guessing-space of backup question answers is obviously too small, or involves questions whose answers can be looked up on the Internet for targeted or popular personalities. Several high-profile break-ins have exploited this fact.

Proposed authentication alternatives exist (e.g., [52]), but require more study. In summary, the implementation of password reset mechanisms is sensitive, fraught with dangers, and may require case-specific decisions.

4.7 Phishing

Guessing is but one means to get a password. Phishing rose to prominence around 2005 as a simple way to socially engineer users into divulging secrets. There are two varieties. Generic or scattershot attempts are generally delivered in large spam campaigns; *spear phishing* aims at specific individuals or organizations, possibly with target-specific lures to increase effectiveness.

Scattershot phishing generally exploits user confusion as to how to distinguish a legitimate web-site from a spoofed version [19]. The literature suggests many approaches to combat the problem, e.g., toolbars, tokens, two-factor schemes, user training. Few of these have enjoyed large-scale deployment. One that did, the SiteKey image to allow a user to verify a site, was found not to meet its design goals [51]: most users entered their password at a spoofed site even in the absence of the trust indicator. A toolbar indicator study reached a similarly pessimistic conclusion [62]. Equally, no evidence suggests any success from efforts to train users to tell good sites from bad simply by parsing the URL; the task itself is ill-defined [29]. In fact, much of the progress against scattershot phishing in recent years appears to have been by browser vendors, through better identification and blocking of phishing sites.

Spear phishing continues to be a major concern, especially for high-consequence sites. The March 2011

breach on RSA Security's SecurID hardware tokens was reportedly[8] such an attack. It is too early to say if approaches wherein administrators send periodic (defensive training) phishing emails to their own users leads to improved outcomes.

4.8 Re-using email address as username

Many sites (over 90% by one study [10]) encourage or force users to use an email address as username. This provides a point of contact (e.g., for password resets—or marketing), ensures unique usernames, and is memorable. However it also brings several security issues.

It encourages users (subconsciously or otherwise) to re-use the email password, thereby increasing the threats based on password re-use [16]. It can facilitate forms of phishing if users become habituated to entering their email passwords at low-value sites that users email addresses as usernames.

Re-using email addresses as usernames across sites also facilitates leaking information regarding registered users of those sites [50], although whether a given string is a valid username at a site can be extracted for non-email address usernames also [10, 11]. Preventing such leaks may be as much a privacy issue, as a security issue.

5 Discussion and implications

5.1 System-side vs. client-side defences

Some password-related defences involve implementation choices between system-side and client-side mechanisms; some attacks can be addressed at either the server (at cost of engineering effort) or the client (often at cost of user effort). Table 6 summarizes costs and benefits of several measures that we have discussed, noting security benefit and usability cost.

We have seen little discussion in the literature of the available trade-offs—and implications on cost, security, usability, and system-wide efficiency with respect to total user effort—between implementing password-related functionality client-side vs. server-side. Ideally, all decisions on where to impose costs would be made explicitly and acknowledged. A danger is that costs offloaded to the user are often hard to measure, and therefore unmeasured—this does not make the cost zero, but makes it hard to distinguish from zero. It is a natural consequence that system-side costs, which are more directly visible and more easily measured, are under-utilized, at the expense of client-side mechanisms which download (less visible, harder to measure) cognitive effort to end-users. For example, forcing users to choose passwords that will resist many guesses is a way of addressing the threat of offline attacks, and relies almost exclusively on user effort. Investing engineering time to better protect the password file, to ensure that leaks are likely to be detected, and to ensure that passwords are properly salted and hashed (or protected using an offline-resistant scheme such as discussed in Section 3.6) are alternatives dealing with the same problem that rely on server-side effort (engineering effort and/or operational time). Florêncio and Herley [23] found that sites where users do not have a choice (such as government and university sites) were more likely to address the offline threat with user effort, while sites that compete for users and traffic (such as retailers) were more likely to allow password policies that addressed the online threat only.

Scale is important in deciding how costs should be divided between the server and client sides; what is reasonable at one scale may be unacceptable at another. For example, many web-sites today have many more accounts than the largest systems of 30 years ago. A trade-off inconveniencing 200 users to save one systems administrator effort might be perfectly reasonable; however, the same trade-off involving 100 million users and 10 administrators is a very different proposition: the factor of $50,000$ increase in the ratio of users to administrators means that decisions should be approached differently, especially in any environment where user time, energy, and effort is a limited resource. There is evidence that the larger web-sites take greater care than smaller ones to reduce the burden placed on users [23].

5.2 Take-away points

We now summarize some of the key findings, and make recommendations based on the analysis above.

Many different types of sites impose passwords on users; asset values related to these sites and associated accounts range widely, including different valuations between users of the same sites. Thus, despite little attention to date in the literature, recognizing different categories of accounts is important (cf. Table 1). User effort available for managing password portfolios is finite [3, 25, 27, 57]. Users should spend less effort on password management issues (e.g., choosing complex passwords) for don't-care and lower consequence accounts, allowing more effort on higher consequence accounts. Password re-use across accounts in different categories is dangerous; a major concern is lower consequence sites compromising passwords re-used for high-consequence sites. While this seems an obvious concern, a first step is greater formal recognition of different categories of sites. We summarize this take-away point as:

T1: *Recognizing different categories of web-sites is essential to responsibly allocating user password*

[8] http://www.wired.com/2011/08/how-rsa-got-hacked/

IMPLEMENTATION ASPECT	ATTACKS STOPPED OR SLOWED	USER IMPACT	REMARKS
Password stored non-plaintext	Full compromise on server breakin alone	None	Recommended
Salting (global and per-account)	Pre-computation attacks (table lookup)	None	Recommended
Iterated hashing	Slows offline guessing proportionally	None	Recommended
MAC of iterated, salted hash	Precludes offline guessing (requires key)	None	Best option (key management)
Rate-limiting & lockout policies	Hugely reduces online guessing	Possible user lockout	Recommended
Blacklisting (proactive checking)	Eliminates most-probable passwords	Minor for small lists	Recommended
Length rules	Slows down naive brute force attacks	Cognitive burden	Recommended: length ≥ 8
Password meters	Nudges users to "less guessable" passwords	Depends on user choice	Marginal gain
Password aging (expiration)	Limits ongoing attacker access; indirectly ameliorates password re-use	Significant; annoying	Possibly more harm than good
Character-set rules	May slow down naive brute-force attacks	Cognitive burden. Slows entry on mobile devices	Often bad return on user effort

Table 6: Password-related implementation options. The majority of Remarks are relevant to medium-consequence accounts (see Table 1). It is strongly recommended that password storage details (e.g., salting, iterated hashing, MAC if used) are implemented by standard library tools.

management effort across sites. Users are best served by effort spent on higher consequence sites, and avoiding cross-category password re-use.

While naive "password strength" measures are widely used, simple to calculate, and have formed the basis for much of the analysis around passwords, simplistic metrics [13] based on Shannon entropy are poor measures of guessing-resistance (recall Section 3.1). Reasoning that uses naive metrics as a proxy for security is unsound and leads to unreliable conclusions. Policies, requirements and advice that seek to improve password security by "increasing entropy" should be disregarded.

T2: *Crude entropy-based estimates are unsuitable for measuring password resistance to guessing attacks; their use should be discouraged.*

While choosing passwords that will resist (online and/or offline) guessing has dominated the advice directed at users, it is worth emphasizing that the success rate of several attacks are unaffected by password choice.

T3: *The success of threats such as client-side malware, phishing, and sniffing unencrypted wireless links are entirely unaffected by password choice.*

Password policies and advice aim to have users choose passwords that will withstand guessing attacks. The threshold number of guesses to survive online and offline attacks differ enormously. The first threshold does not grow as hardware and cracking algorithms improve; the second gradually increases with time, only partially offset by adaptive password hashing functions (if used).

T4: *Password guessing attacks are either online or offline. The guessing-resistance needed to survive the two differs enormously. Withstanding 10^6 guesses probably suffices for online; withstanding 10^{14} or more guesses may be needed to resist determined, well-resourced offline attacks.*

There is no continuum of guessing attack types—it is either online or offline, with nothing in between. There is a chasm between the threshold to withstand these two different types. There is little security benefit in exceeding the online threshold while failing to reach the offline one. Passwords that fail to completely cross this chasm waste effort since they do more than is necessary to withstand online attacks, but still succumb to offline attacks.

T5: *Between the thresholds to resist online and offline attacks, incremental improvement in guess-resistance has little benefit.*

Recall that rainbow table attacks are one form of offline attack, and require access to leaked password hashes.

T6: *Rainbow table attacks can be effectively stopped by well-known salting methods, or by preventing the leakage of hashed password files.*

Analysis of Fig.1 shows that offline attacks are possible and necessary in only very limited circumstances which occur far less often than suggested from the attention given by the research literature. If the password file has not been properly salted and hashed, then user effort to withstand beyond 10^6 guesses is better spent elsewhere.

T7: *Offline guessing attacks are a major concern only if the password file leaks, the leak goes undetected, and the file was properly salted and hashed (otherwise simpler attacks work, e.g., rainbow tables).*

It follows that sites that store passwords in plaintext or reversibly encrypted, and impose strict password composition policies unnecessarily burden users—the policies offer zero benefit against intelligent attackers, as any increased guessing-resistance is irrelevant. The attacker either has direct access to a plaintext password, or if the key encrypting the hashed password does not also leak then the (plaintext) password hashes needed for the offline guessing attack are unavailable.

T8: *For implementations with stored passwords available at the server (plaintext or reversibly encrypted), composition policies aiming to force resistance to offline guessing attacks are unjustifiable—no risk of offline guessing exists.*

The threat of offline guessing attacks can essentially be eliminated if it can be ensured that password files do not leak, e.g., by keyed hash functions with HSM (hardware security) support. Guessing attack risks then reduce to online guessing, which is addressable by known mechanisms such as throttling, recognizing known devices, and proactive checking to disallow too-popular passwords—all burdening users less than composition policies.

T9: *Online attacks are a fact of life for public-facing servers. Offline attacks, by contrast, can be entirely avoided by ensuring the password file does not leak, or mitigated by detecting if it does leak and having a disaster-recovery plan to force a system-wide password reset in that case.*

6 Concluding remarks

In concluding we summarize the case against consuming user effort in attempts to resist offline guessing attacks.

1. Honesty demands a clear acknowledgement that we don't know how to do so: attempts to get users to choose passwords that will resist offline guessing, e.g., by composition policies, advice and strength meters, must largely be judged failures. Such measures may get *some* users across the online-offline chasm, but this helps little unless it is a critical mass; we assume most administrators would consider a site with half its passwords in an attacker's hands to be fully rather than half compromised.

2. Failed attempts ensure a large-scale waste of user effort, since exceeding the online while falling short of the offline threshold delivers no security benefit.

3. The task gets harder every year—hardware advances help attackers more than defenders, increasing the number of guesses in offline attacks.

4. Zero-user-burden mechanisms largely or entirely eliminating offline attacks exist, but are little-used.

5. Demanding passwords that will withstand offline attack is a defense-in-depth approach necessary only when a site has failed both to protect the password file, and to detect the leak and respond suitably.

6. That large providers (e.g., Facebook, Fidelity, Amazon) allow 6-digit PINs demonstrates that it is possible to run first-tier properties without placing the burden of resisting offline attacks on users.

Preventing, detecting and recovering from offline attacks must be administrative priorities, if the burden is not to be met with user effort. It is of prime importance to ensure that password files do not leak (or have content such that leaks are harmless), that any leak can be quickly detected, and that an incident response plan allows system-wide forced password resets if and when needed. Next, and of arguably equal importance, is protecting against online attacks by limiting the number of online guesses that can be made (e.g., by throttling or lockouts) and precluding the most common passwords (e.g., by password blacklists). Salting and iterated hashing are of course expected, using standardized adaptive password hashing functions or related MACs.

Acknowledgements. We thank Michael Brogan and Nathan Dors (U. Washington) for helpful discussions, anonymous referees, and Furkan Alaca, Lujo Bauer, Kemal Bicakci, Joseph Bonneau, Bill Burr, Nicolas Christin, Simson Garfinkel, Peter Gutmann, M. Mannan, Fabian Monrose, and Julie Thorpe for detailed comments on a draft. The third author acknowledges an NSERC Discovery Grant and Canada Research Chair in Authentication and Computer Security.

References

[1] A. Adams and M. A. Sasse. Users Are Not the Enemy. C.ACM, 42(12), 1999.

[2] M. Alsaleh, M. Mannan, and P. C. van Oorschot. Revisiting defenses against large-scale online password guessing attacks. IEEE TDSC, 9(1):128–141, 2012.

[3] A. Beautement and A. Sasse. The economics of user effort in information security. Computer Fraud & Security, pages 8–12, October 2009.

[4] A. Beautement, M. Sasse, and M. Wonham. The Compliance Budget: Managing Security Behaviour in Organisations. In NSPW, 2008.

[5] F. Bergadano, B. Crispo, and G. Ruffo. High dictionary compression for proactive password checking. ACM Trans. Inf. Syst. Secur., 1(1):3–25, 1998.

[6] J. Bonneau. Guessing human-chosen secrets. University of Cambridge. Ph.D. thesis, May 2012.

[7] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In Proc. IEEE Symp. on Security and Privacy, pages 538–552, 2012.

[8] J. Bonneau. Password cracking, part II: when does password cracking matter, Sept.4, 2012. `https://www.lightbluetouchpaper.org`.

[9] J. Bonneau, C. Herley, P. van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Proc. IEEE Symp. on Security and Privacy, 2012.

[10] J. Bonneau and S. Preibusch. The password thicket: Technical and market failures in human authentication on the web. In WEIS, 2010.

[11] A. Bortz and D. Boneh. Exposing private information by timing web applications. Proc. WWW, 2007.

[12] S. Brostoff and M. Sasse. "Ten strikes and you're out": Increasing the number of login attempts can improve password usability. CHI Workshop, 2003.

[13] W. Burr, D. F. Dodson, and W. Polk. Electronic Authentication Guideline. In NIST Special Pub 800-63, 2006.

[14] W. Cheswick. Rethinking passwords. ACM Queue, 10(12):50–56, 2012.

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

[16] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The tangled web of password reuse. NDSS, 2014.

[17] X. de Carnavalet and M. Mannan. From very weak to very strong: Analyzing password-strength meters. In Proc. NDSS, 2014.

[18] S. Designer and S. Marechal. Password Security: Past, Present, Future (with strong bias towards password hashing), December 2012. Slide deck: http://www.openwall.com/presentations/Passwords12-The-Future-Of-Hashing/.

[19] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In Proc. CHI, 2006.

[20] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley. Does my password go up to eleven? the impact of password meters on password selection. In Proc. CHI, 2013.

[21] S. Fahl, M. Harbach, Y. Acar, and M. Smith. On the ecological validity of a password study. In Proc. SOUPS. ACM, 2013.

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

[23] D. Florêncio and C. Herley. Where Do Security Policies Come From? Proc. SOUPS, 2010.

[24] D. Florêncio, C. Herley, and B. Coskun. Do Strong Web Passwords Accomplish Anything? Proc. Usenix Hot Topics in Security, 2007.

[25] D. Florêncio, C. Herley, and P. van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In Proc. USENIX Security, 2014.

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

[27] B. Grawemeyer and H. Johnson. Using and managing multiple passwords: A week to a view. Interacting with Computers, 23(3):256–267, 2011.

[28] E. Grosse and M. Upadhyay. Authentication at scale. IEEE Security & Privacy, 11(1):15–22, 2013.

[29] C. Herley. So Long, And No Thanks for the Externalities: Rational Rejection of Security Advice by Users. Proc. NSPW, 2009.

[30] C. Herley and P. van Oorschot. A research agenda acknowledging the persistence of passwords. IEEE Security & Privacy, 10(1):28–36, 2012.

[31] A. Juels and R. L. Rivest. Honeywords: Making password-cracking detectable. In Proc. ACM CCS, pages 145–160, 2013.

[32] 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. In Proc. IEEE Symp. on Security and Privacy, 2012.

[33] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure applications of low-entropy keys. Proc. ISW'97—Springer LNCS, 1396:121-134, 1998.

[34] M. Mannan, D. Barrera, C. D. Brown, D. Lie, and P. C. van Oorschot. Mercury: Recovering forgotten passwords using personal devices. In Financial Cryptography, pages 315–330, 2011.

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

[36] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[37] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[38] R. Morris and K. Thompson. Password Security: A Case History. C.ACM, 22(11):594–597, 1979.

[39] A. Muller, M. Meucci, E. Keary, and D. Cuthbert, editors. OWASP Testing Guide 4.0. Section 4.5: Authentication Testing (accessed July 27, 2014), https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents.

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

[41] D. Norman. The Way I See It: When security gets in the way. Interactions, 16(6):60–63, 2009.

[42] oclHashcat. http://www.hashcat.net/.

[43] P. Oechslin. Making a faster cryptanalytical time-memory trade-off. Advances in Cryptology - CRYPTO 2003, 2003.

[44] Openwall. http://www.openwall.com/john/.

[45] OWASP. Guide to Authentication. Accessed July 27, 2014, https://www.owasp.org/index.php/Guide_to_Authentication.

[46] C. Percival. Stronger key derivation via sequential memory-hard functions. In BSDCan, 2009.

[47] B. Pinkas and T. Sander. Securing Passwords Against Dictionary Attacks. ACM CCS, 2002.

[48] N. Provos and D. Mazieres. A future-adaptable password scheme. In USENIX Annual Technical Conference, FREENIX Track, pages 81–91, 1999.

[49] R. W. Reeder and S. Schechter. When the password doesn't work: secondary authentication for websites. IEEE Security & Privacy, 9(2):43–49, 2011.

[50] P. F. Roberts. Leaky web sites provide trail of clues about corporate executives. ITworld.com, August 13, 2012.

[51] S. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators: evaluation of website authentication and effect of role playing on usability studies. In Proc. IEEE Symp. on Security and Privacy, 2007.

[52] S. Schechter, S. Egelman, and R. Reeder. It's not what you know, but who you know: a social approach to last-resort authentication. In Proc. CHI, 2009.

[53] S. E. Schechter, A. J. B. Brush, and S. Egelman. It's no secret: Measuring the security and reliability of authentication via "secret" questions. In Proc. IEEE Symp. Security & Privacy, 2009.

[54] Schechter, S. and Herley, C. and Mitzenmacher, M. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. Proc. HotSec, 2010.

[55] E. H. Spafford. OPUS: Preventing weak password choices. Computers & Security, 11(3):273–278, 1992.

[56] J. Steven and J. Manico. Password Storage Cheat Sheet (OWASP). OWASP. Apr.7, 2014, https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet.

[57] E. Stobert and R. Biddle. The password life cycle: user behaviour in managing passwords. In Proc. SOUPS, 2014.

[58] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, et al. How does your password measure up? The effect of strength meters on password creation. In Proc. USENIX Security, 2012.

[59] P. van Oorschot and S. Stubblebine. On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop. ACM TISSEC, 9(3):235–258, 2006.

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

[61] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In Proc. IEEE Symp. on Security and Privacy, pages 391–405, 2009.

[62] M. Wu, R. Miller, and S. L. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks. Proc. CHI, 2006.

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

[64] E. Zwicky. Brute force and ignorance. ;login:, 35(2):51–52, April 2010. USENIX.