Myphrase: Passwords from Your Own Words

Adam Skillen and Mohammad Mannan
{a_skil, mmannan}@ciise.concordia.ca

Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Canada

January 24, 2013

Abstract

To improve manageability and strength of user-chosen passwords, we propose a multi-word password scheme called *Myphrase*. Contrary to the often-repeated but failed policy of banning common words as passwords, we encourage users to use words that are more personal to them—irrespective of the words being too common or esoteric. In Myphrase, a small dictionary is created from user-authored content such as sent emails and blogs. A master passphrase is constructed by randomly selecting words from the dictionary. We propose two variants as a trade-off between security and memorability; in *random sequence*, words are chosen uniformly across the dictionary, and in *connected discourse*, words are tagged using a part-of-speech engine and inserted appropriately into sentence templates. Words in the passphrase are expected to be easily recognizable to users and can be efficiently entered by leveraging the auto-suggest feature. Myphrase is designed to be compatible with both desktop and mobile platforms—a growing requirement for current authentication schemes. We create website-specific passwords from the master passphrase by salting the phrase with the site's domain. To restrict offline attacks on the master passphrase from exposed site passwords, we require the passphrase to be of sufficient length (e.g., 6 words from a 1024-word dictionary, resulting in 60 bits of entropy in the random sequence variant). Entropy calculation for the connected discourse variant is less straightforward. We analyze Myphrase dictionaries and expected entropy of generated passphrases with two datasets: the Enron email corpus, and several popular books from Project Gutenberg. We also evaluate Myphrase using a recently proposed, slightly modified, framework of usability-deployability-security ratings, and seek feedback on our proof-of-concept prototypes available for both desktop and mobile platforms.

1 Introduction and Motivation

To the dismay of security proponents and website administrators, many regular and expert users consistently choose *weak* passwords, such as, 123456, iloveyou, ieee2012, and princess (see e.g., leaked passwords from IEEE.org [11] and Rockyou.com [18]). Currently, common dictionary words and their predictable variants are heavily used as passwords. For users, this is apparently the most sensible choice for password creation and long-term management [15]. To exploit this behavior, offline and online password guessing attacks use common passwords as their starting point. To restrict these attacks, some websites forbid certain *obvious* passwords; see e.g., Twitter [39]. On the other hand, to facilitate user choice, Schechter et al. [34] suggested to place a threshold in the use of popular passwords; users are free to choose any password as long as that password has not been chosen by too many other users of the website. This may reduce the total number of compromised accounts on a given site, but does not protect the individuals that choose weak passwords.

Password entry from constrained input interfaces of mobile devices may further influence users to choose dictionary words as passwords (cf. [20]). Recently, multi-word password schemes have been revisited as an alternative to regular passwords by leveraging auto-correct and auto-suggest features in mobile devices [10, 19]. However, as long as users are free to choose words in a multi-word phrase, no significant improvement is apparent in password strength (see e.g., [23, 8, 43, 35]).

With *Myphrase*, we explore a different approach, which is apparently more in-line with user desire and can support both desktop and mobile platforms. Instead of discouraging users from choosing what they are comfortable with, we leverage users' *own* personal vocabulary to generate strong passwords; i.e., any words can be used irrespective of being considered too *obvious* or *taboo*. A Myphrase passphrase consists of multiple randomly chosen words from a user-created/selected dictionary. The dictionary is user-specific, e.g., generated from user-created text content, such as emails; users may also choose a pre-generated dictionary aligned with their interests e.g., lexicon from a favorite poet. Users can even use a list of common passwords as their dictionary, e.g., the 3546-word John the Ripper most common password list.[1] Dictionaries need not be private.

We examine two variations for passphrase construction. The first variant, called Random Sequence (RS), randomly selects words without regard for syntax. For an expected level of entropy, the required number of words in an RS-passphrase can be easily set; e.g., for 60 bits of entropy, six words must be chosen from a 1024-word dictionary. Memorability of RS-phrases is expected to be benefited from the user's familiarity of the words (i.e., frequency of occurrence/use; cf. [33, 17, 13]). The second variant, called Connected Discourse (CD), constructs proper sentences using rudimentary natural language processing. Dictionary words are tagged using a part-of-speech (POS) engine, and inserted into pre-created sentence templates (cf. Mad Libs [30]). Calculating entropy for this variant is not as straightforward as

[1] http://www.openwall.com/passwords/wordlists/password-2011.lst

the RS variant; entropy depends on the selected sentence template and the breakdown of words in each POS category from the dictionary. Memorability of CD-phrases is expected to be benefited from high-order syntactic and semantic structures (cf. [25, 24]), in addition to word familiarity. To reduce memory load, we expect users to memorize only one Myphrase master passphrase, and use it across multiple websites. A site password is generated by salting the phrase with the site's URL domain; the salted password is also hashed and converted into a server-compatible password (e.g., consisting of alphanumeric characters only).

We test Myphrase by creating dictionaries for users in the Enron email corpus and authors from Project Gutenberg. We find that our frequency ranked dictionaries have similar POS break-down, as compared to text parsed from a large collection of prose. We also compare the similarity between user's dictionaries to determine how personal or unique they are. We used these findings to identify sentence templates that maximize a passphrase's complexity.

In summary, Myphrase offers the following benefits.

1. Passphrase from user-specific words: Myphrase introduces a middle-ground between conflicting choices for text passwords: machine-generated (strong but memory-unfriendly), and user-chosen (memory-friendly but weak). Myphrase phrases are machine generated but consist of a user's personally-meaningful words; these passphrases are expected to be both memory-friendly *and* strong.
2. Stronger passwords: To restrict offline attacks, Myphrase passwords offer a significant entropy gain; e.g., 60 bits in Myphrase's default setting (cf. an estimated 10-20 bits of entropy in current passwords [6]). Passwords are also further strengthened using the PBKDF2 key-stretching function by a factor of 2^{15}. The exact entropy of Myphrase passwords can easily be determined for the RS variant and closely approximated for the CD variant. Entropy of user-chosen passwords can at best be roughly estimated (see e.g., [9, 44]).
3. Scalable and resilient to site-password leaks: Users need to memorize only one Myphrase master passphrase and can use it for all web logins—irrespective of varying levels of site security. If a site-password is leaked from the server-side, the user can update only that password, while maintaining their master passphrase and all other site-passwords.
4. Cross-platform/device compatibility: Auto-suggesting words after typing (or tapping) a couple of characters may reduce the input time of the passphrase; such a feature makes Myphrase suitable for mobile devices with a constrained/touch-based keyboard. We have implemented proof-of-concept prototypes for both desktop and mobile platforms.

Additionally, Myphrase passwords are resilient to phishing attacks. Attackers get a password specific to their phishing domain, instead of the target domain; they also do not receive the master passphrase. These benefits of Myphrase and limitations are explained further in Section 5.

2 Myphrase Description

Below we describe Myphrase, including our assumptions and user steps.

Operational and threat model assumptions. The custom dictionary can be generated from different types of user content, such as: (a) user-authored content, e.g., sent emails, tweeted messages, blog posts, comments at social networking sites, academic papers, and other such documents; and (b) user-liked content, e.g., favorite ebooks, song lyrics, and emails from certain contacts. Alternatively, users may select a pre-generated dictionary according to their interests or familiarity such as: an urban dictionary, medical or technology dictionary. For the connected discourse (CD) variant of Myphrase, the dictionary must contain the following POS categories: nouns, verbs, adjectives and adverbs. However, the random sequence (RS) variant has no such constraints (e.g., the dictionary may be comprised entirely of nouns, as in names of cities or film actors). The dictionary itself and sources of the dictionary words are not security-sensitive, and can be made public. However, the dictionary may be privacy-sensitive, if users do not want to reveal that certain words appear in their dictionary.

We assume that words in the created/selected personal dictionary are familiar enough to users that they can memorize a relatively long sequence (e.g., six) of these words as their master passphrase; see Appendix B for a discussion on memorability. To make use of auto-suggest, for minimal typing and easy recognition, Myphrase requires the dictionary be available in all user devices (e.g., via manual copy, email attachment, browser sync mechanisms or web hosting). The dictionary can be re-created from previously selected sources. Similar to regular passwords, the Myphrase master passphrase is vulnerable to host malware and shoulder-surfing attacks; such attacks are out-of-scope.

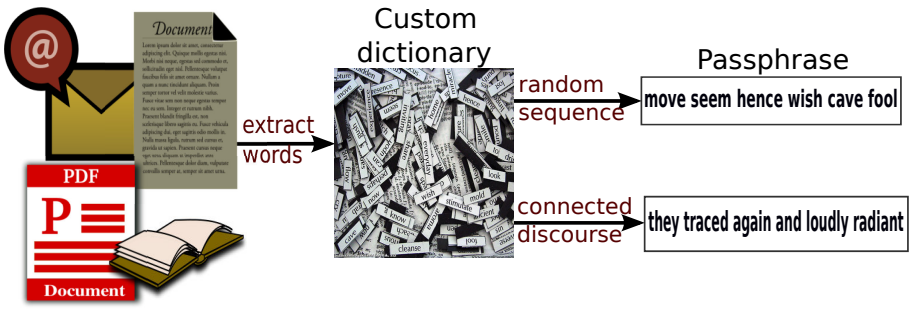

Figure 1: *Myphrase basic mechanism with RS and CD variants*

Myphrase design. When constructing the user's dictionary, we rank words by occurrence and keep only the most frequently used words. During this operation, we omit 215 common conjunctions, articles, prepositions, and pronouns (e.g., 'at', 'and', 'she') from the frequency ranking, and append them to the dictionary before passphrase generation. These words are necessary parts-of-speech, and must be present regardless of their observed frequencies

(for the CD variant). Conversely, these parts-of-speech tend to be over-represented in a user's dictionary, as compared to the POS breakdown observed in a large collection of prose. Removing these words from the frequency ranking should allow us to capture more personally meaningful, and hopefully memorable, words from the input. This also allows us to more accurately estimate the entropy for a given CD passphrase; see Section 5.1. We experimented with different lists of common words ranging from 100 to 1000. We found that 215 was the optimal size to ensure enough variation was available for the CD sentence templates, and balance dictionary POS break-down and passphrase entropy.

In Myphrase, passphrases are generated by randomly selecting words from the user's vocabulary in the following ways. (*i*) Random sequence: n words are randomly selected from the user's dictionary. (*ii*) Connected discourse: words are classified based on their POS tag (e.g., verb, noun); an n word-long sentence template is randomly selected from a pool of pre-created templates; dictionary words are then randomly chosen for each position in the template, based on their POS class.[2] The value of n depends on the dictionary size and expected level of entropy. Currently, we recommend that the passphrase offer at least 60 bits of entropy to restrict offline dictionary attacks; e.g., $n = 6$ for a 1024-word dictionary for the RS variant. The entropy for a CD passphrase cannot be directly calculated. Through experimentation we found that the complexity of a CD-phrase is roughly 65% of an equivalent length RS-phrase, so $n = 8$ words from a 4096-word dictionary is recommended; see Section 5.1 for details.

Specific words in the generated passphrase can be indicated for replacement, and Myphrase will offer another random word. However, entropy of the phrase may suffer, depending on the dictionary size and the number of iterations used for a specific word. We limit the loss of entropy by restricting the number of times a user may selectively regenerate words in the phrase to n, e.g., 8 times for an 8 word passphrase; see Appendix A.

The master passphrase is used to generate unique site-specific passwords as follows: $pwd = \text{Hash2Text}(h^i(passphrase||domain||updatecount))$; h is a cryptographic hash/key-stretching function, e.g., SHA-1 or PBKDF2; i is the number of hash iterations e.g., $i = 32768$; and *updatecount* is the number of updates made to a specific site's password (by default $updatecount = 0$, see also "Updating site passwords" below). A higher value of i will make brute-force attacks on the passphrase more computationally intensive, but values that are too high may slow down password generation. To increase the cost of offline cracking attacks, we use PBKDF2 in our implementation, which is more complex than SHA-1 alone, when implemented in a custom circuit. Recently proposed computational and memory-wise expensive functions can also be used (e.g., [22, 28]). The Hash2Text function encodes the binary hash result into a server-compatible password, e.g., passwords with only alphanumeric characters; cf. PwdHash [32].

[2]This variant resembles the popular word game Mad Libs [30], in which players in turn choose words of particular types to fill-in the blanks of a given sentence template. However, there are no fixed words in our templates, and all words are chosen randomly from a user's vocabulary.

User steps: dictionary and master passphrase generation. The following steps are required to generate the master passphrase; see Fig. 1. **(*a*)** Users first select word sources for their custom word dictionary, e.g., emails, and ebooks. **(*b*)** The Myphrase tool extracts words from these sources, ranks the words by frequency, and selects a pre-specified number (e.g., 2048) of the most frequent words as the dictionary. After creation, users may choose to manually update the words in their dictionary. **(*c*)** The tool creates a multi-word passphrase from the dictionary created in the previous step (or a pre-selected dictionary) using the RS or CD variant. **(*d*)** Users can continue generating new phrases until they are satisfied. When the user is satisfied with a passphrase, she is expected to memorize the word sequence verbatim, or to write it down in a *secure* place, (cf. [48, 16]). The exact spelling of words need not be memorized due to the auto-suggest feature of Myphrase. However, users must not rearrange the words in a phrase, as such modifications may reduce effective entropy.

User steps: site-specific password generation. **(*a*)** Users enter their passphrase; assuming the dictionary is available as a browser add-on or with the Myphrase application, users need to type only the first couple of characters for each word and then select the correct word from the auto-suggestion list. **(*b*)** The site-specific password is created using the formula shown above. The site password is sent to the authenticating site.

Updating site passwords. Users may want to update their Myphrase site passwords for various reasons, including a periodic password update policy, and passwords compromised at server-side (e.g., LinkedIn's 6.5M password leak [3]). We allow password update without requiring the master passphrase to be changed, by increasing the value of *updatecount* (see the above site-password generation formula). This value is site-specific and starts from zero; when a site's password needs to be updated, *updatecount* is incremented by one, and the updated value is used only for the target site (i.e., other site passwords will still be generated with $updatecount = 0$, if not already changed). This solution is similar to the index mechanism as proposed by Halderman et al. [14]. Site URLs with updated count values must be stored and synced across devices; e.g., via Firefox Sync.

3 Implementation

We implemented Myphrase for PCs as a Firefox addon, and for Android devices as a custom soft-keyboard. The desktop version provides an interface to build dictionaries, generate passphrases, and insert site passwords. The mobile version currently makes use of dictionaries and passphrases generated in the desktop version to insert passwords into websites. A web interface is also created to facilitate password generation when other tools are unavailable (e.g., a temporary device). We currently handle only English words; internationalization would require POS taggers for additional languages. Details are discussed below.

Preferences. The Firefox addon offers a few user customizable settings. We provide some pre-built dictionaries, but encourage users to create personalized ones. To help guide the user experience, we provide the following default options: (*a*) type of master passphrase: connected discourse; (*b*) dictionary: lexicon from the works of H. G. Wells; (*c*) length of master passphrase: eight words; (*d*) size of custom dictionary: 4096 words; and (*e*) maximum word length: 14 characters. The addon also has site-specific settings for each generated site password. These allow Myphrase to conform to provider specific constraints, and store each site's password update count. The defaults are as follows: (*a*) length of site passwords: 12; (*b*) special characters: disabled; and (*c*) password update count: 0.

Dictionary construction. The Firefox addon can build a personal dictionary from the user's outbound emails, and plain-text, HTML and XML files. Our text and HTML/XML parsers can be used to build dictionaries from PDF and other document formats after saving as text, ebooks (e.g., ePub, a compressed XML format), and web content saved as HTML. For email parsing, we choose to collect messages from the Simple Mail Firefox addon [40], which enables managing several email accounts, including POP/IMAP exposed webmails. It aggregates emails across accounts in a single folder structure (i.e., emails sent from all accounts are stored in one "Sent" folder). This allows us to gather words that may originate from disparate vocabularies (e.g., work emails with professional terms vs. personal emails with slang or informal words). Sent messages are collected by querying an SQLite database. The message text is parsed to extract the user's original text; i.e., we eliminate quoted text from forwarded or replied emails.

Using a series of regular expressions, we decode HTML/XML entities and eliminate mark-up, punctuation, digits, and other non-word strings. By default, we capture strings that are 14 characters or less. Note that short words do not compromise the security of Myphrase; however, longer words may require more typing if the user's dictionary is not available for auto-suggest completion.

We retain the top 4096 words in the user's dictionary by default. All characters are reduced to lower-case before the word frequency analysis. The dictionary is saved as a text file, which the user may further customize (manually). Users may also choose to use a pre-built dictionary; we provide a few default dictionaries created using top free ebooks from Project Gutenberg.

Master passphrase generation. After selecting a dictionary, users can generate a passphrase in the Firefox addon. To select random words from the dictionary, we use Mozilla's PRNG, *nsIRandomGenerator* (assumed to be cryptographically secure; uses mouse movement events within the Firefox window). For the RS variant, a passphrase is generated simply by selecting n random words from the dictionary. For the CD variant, we first randomly choose a pre-built sentence template for the given passphrase length ($n = 4$ to 12 words). A number of templates were created by parsing top free ebooks from Project Gutenberg, and selecting a few which appear to yield high entropy passphrases (discussed more in Section 5.1). We

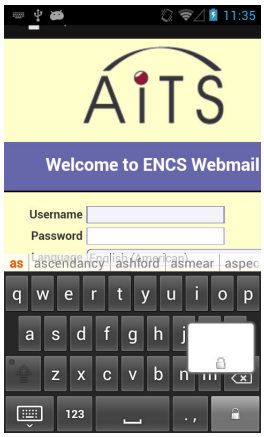

Figure 2: *Myphrase Mobile Keyboard*

then use a Javascript POS tagger [42] to classify the dictionary words. A passphrase is generated by filling the template with randomly selected words from the classes that appear in the template. For both variants, users can regenerate new phrases until they are satisfied; a limited number of selective regeneration of words in a selected phrase is also allowed. The master passphrase is not stored within our software, and cannot be regenerated; users are expected to memorize it.

Site password generation. We anticipate that users will build personal dictionaries and generate passphrases only occasionally. The day-to-day use of Myphrase will largely consist of deriving site passwords. When faced with a login page, the user right-clicks on the password field and selects the Myphrase insert option from the browser context menu. On a mobile device, the user can pull down the notification bar, to switch from their default keyboard to the Myphrase keyboard; see Fig. 2.

The user is then prompted to enter her master passphrase. The prompt allows the user to view the passphrase text, or have it shadowed in the case she is in a public place. An auto-complete suggestion list is populated with words from the user's dictionary to speed up this task. This feature is especially useful on mobile devices where typing tends to be slower. Passphrases are constructed in such a way that the user will not require any modifiers (e.g., shift) or changing keyboard views (e.g., numbers, special characters) which should greatly increase entry time for mobile devices. At this stage, the user can also set site-specific constraints (e.g., password length) and increment the update counter. The update count and constraints for each domain are saved as site specific preference within Firefox and Android, so will only need to be set by the user once.

We then iterate the PBKDF2_HMAC_SHA-1 function 32768 times, using the passphrase as the HMAC secret, and the site's domain and update count as the salt. The choice of our iteration count is explained in Appendix A. The result is then encoded into a compatible password and inserted in the password field. Note that, both the addon and Android soft-

keyboard make only the site-password available to the authenticating domain, which could be a legitimate or phishing site. The master passphrase remains inaccessible to all websites.

Myphrase mobile. We implemented Myphrase for Android devices as a custom soft-keyboard. This would enable Myphrase authentication for any password field on the device including websites, apps and even device unlock; the current in-progress implementation supports only website login. Users can quickly switch to the Myphrase keyboard from the Android notification bar to enter a password. The user's text is displayed while they are typing, but not entered into the password field until they press Enter. As with the desktop software, auto-complete dictionary suggestions are displayed to speed up passphrase entry.

4 Related Work

Multi-word passwords and several-related variants including first-letter mnemonics have been proposed decades ago (see e.g., [2, 29, 23]). Similar schemes have been recently revived to increase password entropy,[3] and to make password input more user-friendly in devices with touch-screen keypads [10, 19]. The use of readily available auto-correct and auto-suggest features can significantly reduce input issues in these devices (e.g., compared to inputting a password with mixed case letters and special characters). Generally, there are two types of multi-word passwords: words and their sequence chosen by a user (e.g., [29, 19]), and words selected randomly from a fixed, system-chosen dictionary (e.g., Cheswick [10]). Cheswick's proposal uses a 1020-word dictionary of *iPhone-friendly* English words. Diceware [31] is another random passphrase generator using one or more dice as the random number generator. Each word in the phrase is chosen from a five digit number generated by five rolls of a die. Any list of 7776 (6^5) unique words can be used; word lists are currently available in several languages.

Smith proposed a word association authentication scheme [38] that requires users to register a list of (cue, response) pairs consisting of personally-meaningful words; obvious pairs such as (black, white) are disallowed. During login, users must provide correct responses to cues chosen randomly by the system. As cues and responses are user-selected and likely to vary significantly from user to user, Smith argues that this scheme would offer adequate benefits in terms of user-acceptance, memorability and security.

The use of natural language processing techniques to create multi-word passwords has also been explored. Atallah et al. [1] generate a meaningful/humorous phrase and associated mnemonic for authentication. Another technique [21] creates sentences for a given random password. News headlines are used as templates by substituting similar words; several variants of the same sentence are generated using a POS tagger and WordNet [27]. To generate multiple passwords from a master mnemonic sentence, Topkara et al. [41] propose

[3]For example, see the popular XKCD cartoon at http://xkcd.com/936/.

to split a password into two parts: a memorized mnemonic sentence and another part written down on paper.

User-chosen phrases or word sequences are almost as memorable as regular passwords [46]; it has long been known that memorability of a sequence of items such as words or phonemes is dependent on familiarity of those items [26, 17, 13]. However, such phrases do not offer much improvement in terms of entropy as users generally choose common phrases; see e.g., Kuo et al. [23]. Another recent analysis of over $100,000$ possible user-selected passphrases from the Amazon PayPhrase system also reported similar results [8]. Generic attack techniques against multi-word passwords have also been proposed; see e.g., Weir [43], Schmitz [35]. In contrast, random words from a system-chosen dictionary offer better entropy, but may not be ideal in terms of memorability. We allow users to reuse the same passphrase safely for all web logins, reducing their memory load and combating the multiple password interference problem.

5 Comparison and Evaluation

In this section, we discuss entropy of passphrases for both RS- and CD-Myphrase variants, memorability of the passphrases and limitations of the scheme. We also use a slightly modified version of the recently-proposed UDS (usability, deployability, security) framework [7] for an analytical evaluation of Myphrase.

5.1 Myphrase Entropy Estimation

Equal-length CD-phrases will provide a lower entropy count than RS-phrases, as each template element is being chosen from a subset of the dictionary words. Experimental entropy estimation of CD-phrases is discussed below. We also allow selective regeneration of words in a phrase. By keeping the regeneration count small, we can limit the loss of entropy. We provide an analysis of selective regeneration and estimated efforts required to launch brute-force guessing attacks against Myphrase passwords in Appendix A.

By default, we expect users to choose six words from a 1024-word dictionary for the RS variant (providing 60 bits of entropy), and eight words from a 4096-word dictionary for the CD variant (providing 60 bits of entropy on average). For easy memorization, users may want to reduce the word count, especially for the RS variant (e.g., three words). This may still provide higher entropy than current passwords (cf. 10-20 bits of entropy as found in a large-scale password study [6]). However, such a three-word master password can be practically retrieved from a leaked site password; this may mandate changing all site passwords. If the adversary has the user dictionary, he can attempt all 3 word combinations until he finds a match with the leaked site password. On average, this will require 2^{29} attempts. As high-impact password leaks happen not so rarely (e.g., [11, 3, 37]), we suggest users to invest

	Composition %			
	Gutenberg authors		Enron authors	
	Avg	Stdev	Avg	Stdev
Nouns	50.29	1.94	63.31	2.84
Verbs	29.27	1.77	22.68	1.70
Adjectives	14.19	0.98	9.16	0.85
Adverbs	6.06	0.60	4.44	0.45
Others	0.19	0.12	0.41	0.08

Table 1: *Frequency adjusted POS classification for 4096-word popular Project Gutenberg author dictionaries (N=10) and Enron dictionaries (N=15)*

in a longer master passphrase. Creating a larger dictionary will also increase the effective complexity of the passphrase.

Experimental entropy estimation of the CD-Myphrase variant. We obtain real world entropy estimates of CD-phrases by examining the POS breakdown for two datasets: Enron emails (from www.cs.cmu.edu/~enron/) and Project Gutenberg ebooks. We also test similarity between user dictionaries.

To generate sentence templates, we parse the 25 most popular ebooks from Project Gutenberg between the period of Aug. 29–Sept. 28, 2012. We identify 60,921 unique strings. The POS class breakdown is as follows: nouns (71%), verbs (18%), adjectives (7%), adverbs (3%), and others (1%). We then select sentence templates from the parsed text, capitalizing on sentences that contained more of the highly populated POS classes. We originally chose ten templates for each possible passphrase length (4 to 12 words) that we believed would yield high entropy phrases; after evaluation, we retain the 7 highest performing templates.

When constructing a user's dictionary we rank words by frequency and select only the most common words in the user's vocabulary. This could skew the user's POS breakdown in such a way that our selected templates would not produce ideal passphrases (e.g., if the frequency ranked dictionaries contain more verbs than nouns). We chose to analyze the Myphrase dictionaries for selected ebook authors and Enron employee emails. For ebooks, we choose 7–10 well-known works from ten authors. The Enron email corpus contains 150 unique user accounts. We isolate each user's personal vocabulary by parsing only sent emails after eliminating quoted text, signature blocks, etc. There were only 15 users with dictionaries of at least 4096 words (we also discarded 2 dictionaries containing several anomalies, e.g., mid-word line-breaks). We found both the Enron employees and ebook authors had similar POS breakdowns in their frequency adjusted dictionaries, validating our template selections; see Table 1. We also noticed earlier that some parts-of-speech (e.g., conjunctions, pronouns) were over-represented; consequently, we remove the 215 common words from those classes before frequency ranking the list.

We then used the created Myphrase dictionaries to calculate expected entropy from each template. Entropy for RS passphrases can be directly calculated, e.g., six words from a

	Entropy count in bits							
	Gutenberg authors				Enron authors			
Length	Min	Max	Avg	Stdev	Min	Max	Avg	Stdev
6 words	41.5	57.5	47.8	5.4	41.2	57.0	47.0	5.4
7 words	47.7	62.6	54.9	4.7	46.5	61.5	53.7	4.4
8 words	58.1	71.0	63.9	3.9	57.2	71.2	63.1	4.4

Table 2: *Observed entropy for phrases from 4096-word Gutenberg and Enron dictionaries. Large deviation is a result of different templates. Each template has little variability as seen in Table 3.*

1024-word dictionary will result in: $log_2(1024^6) = 60$ bits. The entropy of a CD passphrase depends on the template and POS breakdown of the user's dictionary. For example, assume the chosen template is: "noun verb adverb determiner noun verb preposition noun" with POS class sizes (noun, 1975), (verb, 1209), (adverb, 86), (determiner, 21), (preposition, 86). A randomly generated passphrase from this template will provide an entropy of $log_2(1975 \times 1209 \times 86 \times 21 \times 1975 \times 1209 \times 86 \times 1975) = 70.57$. We performed the calculations on 6, 7, and 8 word phrases–see Table 2; on average, these CD phrases retain about 65% of the entropy compared to RS-phrases of respective lengths. The large deviations are a result of the different templates within a given phrase length. The templates provided similar results regardless of the dictionary used. This would suggest that we can further narrow the entropy estimate for a given phrase length by adjusting the templates rather than the dictionary. For example, template five for eight word phrases performed the worst, and could be eliminated. The results of 8-word templates are shown in Table 3.

Template	Avg	Stdev	% of RS
1	71	0.17	72
2	66	0.32	68
3	66	0.12	68
4	61	0.83	63
5	58	0.44	60
6	63	0.64	66
7	60	0.94	62

Table 3: *Observed entropy for 8-word templates; combined results from both Enron and Gutenberg authors (N=25).*

Dictionary uniqueness. We also compared the similarity between the dictionaries to measure how unique they were to each user. For every pair of user dictionaries in both Gutenberg and Enron datasets, we calculated the Jaccard index (i.e., the size of the intersection between two dictionaries divided by the size of their union); see Table 4. The results show that each user dictionary is relatively personal (the average similarity is between 31-43%).

	Jaccard index %			
	Min	Max	Avg	Stdev
Gutenberg	34.84	49.59	42.58	3.62
Enron	23.38	41.82	30.52	3.48

Table 4: *Similarity of user dictionaries (measured as Jaccard index of pairs of users from both Gutenberg and Enron datasets)*

5.2 UDS Evaluation of Myphrase

We now provide an analytical evaluation of Myphrase using the recently-proposed UDS (usability, deployability, security) framework [7]. We modified the ratings slightly, from the original three point scale, to include a fourth *partial-benefit* rating. The *partial-benefit* rating exists between the original *no-benefit* and *quasi-benefit* ratings, and indicates that a given benefit is weakly, or only partially, met. The *quasi-benefit* rating still indicates that a given benefit is almost fully met. For context, we also include regular user-chosen passwords in the UDS evaluation. Due to space limitation, we refer readers to the UDS paper [7] for details of the framework and feature definitions. We have not conducted any formal user-testing yet; to help better design such tests, we would like to get expert feedback and comments on our publicly available prototype. Thus we would like to emphasize that our usability ratings for Myphrase within the UDS framework are only best guesses, given the lack of empirical data on regular users at this point. See Table 5 for the summary of our evaluation. We also provide a brief discussion of our ratings.

Myphrase UDS Ratings Explanation.

We use *Quasi* to refer to "almost full benefit" and *Partially* to indicate "partial benefit only." We rate Myphrase as *Partially-Memorywise-Effortless*/U1: users must remember at least one secret; *Scalable-for-Users*/U2: site-specific passwords are generated from the master passphrase; *Quasi-Nothing-to-Carry*/U3: having the dictionary aids usability (less typing and less error in typing), but passwords can be generated from memorized passphrases; we do not grant *Physically-Effortless*/U4 since, if the dictionary is unavailable, the user is likely to type more characters (for most passphrases) than a regular password; Myphrase is *Partially-Efficient-to-Use*/U6: typing the same passphrase should become easier with repeated use, however, unlike a traditional password manager, the master passphrase must be entered for each authentication; *Quasi-Infrequent-Errors*/U7: use of the same passphrase and auto-fill words from a drop-down menu may result in less typing errors; *Partially-Easy-Recovery-from-Loss*/U8: losing the master secret requires re-setting all site passwords, although the user can browse their dictionary in an attempt to jog their memory; *Partially-Browser-Compatible*/D4: the web interface can be used when software tools are unavailable; *Non-Proprietary*/D6: no known patents as we are aware of. Myphrase is not *Resilient-to-Physical-Observation*/S1: all password input techniques are vulnerable to

	Usability								Deployability						Security									
	U1: Memorywise-Effortless	U2: Scalable-for-Users	U3: Nothing-to-Carry	U4: Physically-Effortless	U5: Easy-to-Learn	U6: Efficient-to-Use	U7: Infrequent-Errors	U8: Easy-Recovery-from-Loss	D1: Accessible	D2: Negligible-Cost-per-User	D3: Server-Compatible	D4: Browser-Compatible	D5: Mature	D6: Non-Proprietary	S1: Resilient-to-Physical-Observation	S2: Resilient-to-Targeted-Impersonation	S3: Resilient-to-Throttled-Guessing	S4: Resilient-to-Unthrottled-Guessing	S5: Resilient-to-Internal-Observation	S6: Resilient-to-Leaks-from-Other-Verifiers	S7: Resilient-to-Phishing	S8: Resilient-to-Theft	S9: No-Trusted-Third-Party	S10: Requiring-Explicit-Consent	S11: Unlinkable
Text passwords			●		●	●	◐	●	●	●	●	●	●	●		◐						●	●	●	●
Myphrase	○	●	◐		●	○	◐	○	●	●	●	○		●		●	●	◐		●	●	●	●	●	●
Fastwords [19]			●		●	○	◐	●	●	●	○					◐						●	●	●	●
Cheswick [10]			◐		●	○	◐	●	●	●	○			●		●	●	●				●	●	●	●

Table 5: *UDS evaluation of Myphrase. Key:* ● *(offers the benefit);* ◐ *(almost offers the benefit);* ○ *(offers partial benefit); blank (benefit not offered).*

physical observation, and inspecting the user's auto-complete selections may allow an adversary to learn the master secret more easily; is *Resilient-to-Targeted-Impersonation*/S2: words in the passphrase are chosen randomly—so having access to a user's preference to certain words or even the user dictionary will not help in targeted guessing; *Resilient-to-Throttled-Guessing*/S3 and *Quasi-Resilient-to-Unthrottled-Guessing*/S4: assuming at least 60 bits of entropy, the generated site-specific password is resilient against online guessing and to some extent, against offline attacks; *Resilient-to-Leaks-from-Other-Verifiers*/S6 and *Resilient-to-Phishing*/S7: each password is site-specific (i.e., salted by the site's domain) and retrieving the master passphrase from a compromised password requires non-trivial computation power (on average, 2^{59} password trials in the default setting; additionally each trial needs 2^{15} iterations of a hash function). This feature also restricts malicious sites from replaying a user's password to get access to another web account of the user (perhaps to a more valuable account).

Differences with Fastwords and Cheswick's scheme. We include Fastwords [19] as an example of multi-word passwords where words are user-chosen. In contrast, Cheswick's scheme [10] generates a passphrase by randomly selecting words from a fixed dictionary. We rate both not offering U1 and U2: they require users to remember several phrases (similar to regular passwords). Fastwords are independent of any particular dictionary (except the

widely-available built-in English dictionary in current mobile platforms); we rate it to offer U3. Cheswick's scheme [10] is rated *Quasi*-U3: similar to Myphrase, the fixed dictionary helps easy recall and typing. We rate both offering *Quasi*-D3: they may require server-side changes as many websites currently disallow the space character in a password. Fastwords do not offer D6: as mentioned at fastword.me, the technology is patent-pending. Fastwords' security features are rated similar to text passwords; we believe it is unlikely to achieve significant improvement in this area, as long as user-choice is involved (cf. [8]).

5.3 Memorability and Limitations

We believe the memorability of a Myphrase passphrase is enhanced by the following factories. (See also Appendix B for a discussion on few studies from psychology.) (*a*) Frequent repetition: repeated use reduces the user's working memory load from potentially dozens of site passwords to one. (*b*) Semantic and syntactic structures: words are apparently more memorable than random character strings, and sentences more memorable than random sequences of words. (*c*) Personally meaningful words: familiarity with the passphrase components should make recall easier. (*d*) Recognition of words: the auto-complete suggestions allow the user to recognize their passphrase words from a list.

Major limitations include: **(*a*)** Myphrase's approach of using a master secret is similar to several existing techniques—expecting that users will remember one strong secret and derive all other site passwords from it. However, users most likely would not change all their existing passwords to Myphrase at the outset (cf. [4]). Therefore, the Myphrase passphrase would be "one more secret" to remember and will benefit users only in the long-run. However, users can gradually migrate their accounts under a Myphrase password, and keep using regular text passwords along with Myphrase. **(*b*)** A forgotten or compromised (e.g., via PC malware) Myphrase master passphrase will incur selecting a new passphrase and resetting all site passwords—a major inconvenience for users. Users may write down the passphrase and store it in a place not accessible to others (cf. [48, 16]). Users may browse the dictionary to attempt to recognize the forgotten words in the passphrase. **(*c*)** The dictionary may be lost or unavailable to users (e.g., when using a new device). In such cases, typing errors may increase as users must recall the exact words in the phrase without any cue. Posting the dictionary to a public or semi-public website (e.g., Facebook) may enable access-from-anywhere. The dictionary may also be re-created from the original sources used. **(*d*)** The Myphrase tool is required to convert the master passphrase to a site password. When the tool is unavailable (e.g., in a friend's device), a website for this conversion is available at: http://users.encs.concordia.ca/~a_skil/myphrase/myp-web/. The web tool uses locally-executed JavaScript, and does not interface with any 3rd party web services. The user may even download the script and execute it locally (offline or even in a virtual machine sandbox). The user enters their passphrase and the URL of the site they wish to log into. The site password is then generated, and the user can copy–paste the site password into the login field

of the web service. **(*e*)** We anticipate the use of Myphrase may increase the login time (e.g., we observed the time to be close to 20 seconds for a 8-word phrase from our own experience on a PC–a formal user study will result in a more accurate estimation). We expect that the repeated use of the same passphrase may reduce the login time in the long-run.

6 Concluding Remarks

Myphrase takes advantage of the already existing tendency towards choosing familiar words as passwords. Users are generally frowned upon by security advocates for making such choices, as these words can easily be subjected to dictionary attacks. In contrast, Myphrase allows users to generate stronger passwords from a dictionary of words they are familiar with or use in their daily communications. To restrict online-only attacks, Florêncio et al. [12] argue that a low-entropy (e.g., 20 bits) password should be enough. However, we believe that in addition to obvious use-cases (e.g., high-value accounts and data encryption), stronger site-specific passwords as generated in Myphrase, are still essential to address problems related to password leakage from websites. Currently, users must change a leaked password for the original site from where it leaked and any other sites that the password is reused; users may not properly keep track of such reuse and may even remain unaware of the compromise unless they follow security news. Note that, storing hashed passwords at the server-side may not always prevent password exposure (e.g., [11, 3]). We believe that risks from leaked passwords are significant due to password reuse, and changing several site passwords as a consequence of one site being compromised is a major inconvenience for users (assuming they would actually update those passwords). We argue that site password leakage should be seriously considered in designing new password schemes, as such incidents are becoming more commonplace.

As discussed, Myphrase has several potential limitations, e.g., longer login times, and memorizing a sequence of several words as the master passphrase. However, the use of personal words may help user-acceptance[4] — a major obstacle for any new password scheme. The auto-suggest feature reduces typing, which may also make Myphrase more suitable for mobile devices than regular passwords. However, we would like to emphasize that no formal user-testing has been conducted yet. We introduce Myphrase here to promote discussion on authentication schemes that can sustain site-password leakage and are suitable for both desktop and mobile platforms. Our prototype implementation is available at:
http://users.encs.concordia.ca/~a_skil/myphrase/.

[4]See e.g., the user study [5] of object-based password: users browse their personal images/music files as part of the login mechanism; some users reportedly *enjoyed* interacting with such objects.

References

[1] Mikhail J. Atallah, Craig J. McDonough, Victor Raskin, and Sergei Nirenburg. Natural language processing for information assurance and security: an overview and implementations. In *NSPW'00*, Ballycotton, County Cork, Ireland, 2000.

[2] Ben F. Barton and Marthalee S. Barton. User-friendly password methods for computer-mediated information systems. *Computers and Security*, 3(3):186–195, August 1984.

[3] BBC News. LinkedIn passwords leaked by hackers. News article (June 7, 2012). http://www.bbc.co.uk/news/technology-18338956.

[4] Kemal Bicakci, Nart Bedin Atalay, Mustafa Yuceel, and Paul C. van Oorschot. Exploration and field study of a browser-based password manager using icon-based passwords. In *Workshop on Real-Life Cryptographic Protocols and Standardization*, March 2011.

[5] R. Biddle, M. Mannan, P.C. van Oorschot, and T. Whalen. User study, analysis, and usable security of passwords based on digital objects. *IEEE TIFS*, 6(3):970–979, September 2011.

[6] Joseph Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symp. on Security and Privacy*, May 2012.

[7] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symp. on Security and Privacy*, May 2012.

[8] Joseph Bonneau and Ekaterina Shutova. Linguistic properties of multi-word passphrases. In *Workshop on Usable Security (USEC'12)*, Bonaire, Netherlands, March 2012.

[9] William Burr, Donna Dodson, and W. Polk. Electronic authentication guidelines (NIST SP 800-63), April 2006.

[10] William Cheswick. Rethinking passwords. Invited talk at USENIX LISA 2010. http://www.usenix.org/event/lisa10/tech/slides/cheswick.pdf. See summary in ;login: The USENIX Magazine, 36(2):68-69, Apr. 2011.

[11] Radu Dragusin. Data breach at IEEE.org: 100k plaintext passwords. Online article (Sept. 18, 2012). http://ieeelog.com/.

[12] Dinei Florêncio, Cormac Herley, and Baris Coskun. Do strong web passwords accomplish anything? In *USENIX Workshop on Hot Topics in Security (HotSec'07)*, Boston, MA, USA, August 2007.

[13] Vernon H. Gregg. *Recall and Recognition*, chapter Word Frequency, Recognition, and Recall. John Wiley & Sons, Inc., 1976.

[14] J. Alex Halderman, Brent Waters, and Edward W. Felten. A convenient method for securely managing passwords. In *Conference on World Wide Web (WWW'05)*, May 2005.

[15] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *NSPW'09*, Oxford, UK, September 2009.

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

[17] Charles Hulme, Sarah Maughan, and Gordon D.A Brown. Memory for familiar and unfamiliar words: Evidence for a long-term memory contribution to short-term memory span. *Journal of Memory and Language*, 30(6):685–701, 1991.

[18] Imperva.com. Consumer password worst practices. Imperva white paper on Rockyou.com's 32 million leaked passwords (Jan. 2010). http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf.

[19] Markus Jakobsson and Ruj Akavipat. Rethinking passwords to adapt to constrained keyboards. In *Mobile Security Technologies (MoST) Workshop*, May 2012.

[20] Markus Jakobsson, Elaine Shi, Philippe Golle, and Richard Chow. Implicit authentication for mobile devices. In *USENIX HotSec'09*, Montreal, Canada, August 2009.

[21] S. Jeyaraman and U. Topkara. Have the cake and eat it too - infusing usability into text-password based authentication systems. In *ACSAC'05*, 2005.

[22] Hugo Krawczyk. Cryptographic extraction and key derivation: the HKDF scheme. In *Crypto'10*, Santa Barbara, CA, USA, 2010. Also published as RFC 5869.

[23] Cynthia Kuo, Sasha Romanosky, and Lorrie Faith Cranor. Human selection of mnemonic phrase-based passwords. In *SOUPS'06*, Pittsburgh, PA, USA, July 2006.

[24] Linda Lombardi and Mary C Potter. The regeneration of syntax in short term memory. *Journal of Memory and Language*, 31(6):713 – 733, 1992.

[25] G. Miller. Human memory and the storage of information. *Information Theory, IRE Transactions on*, 2(3):129 –137, September 1956.

[26] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, March 1956.

[27] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[28] Colin Percival. Stronger key derivation via sequential memory-hard functions. In *BSD Conference (BSDCan'09)*, Ottawa, Canada, 2009.

[29] Sigmund N. Porter. A password extension for improved human factors. *Computers and Security*, 1(1):54–56, January 1982.

[30] Roger Price and Leonard Stern. *The Original Mad Libs 1.* Price Stern Sloan, February 1974.

[31] Arnold Reinhold. Diceware passphrase. http://world.std.com/~reinhold/diceware.html.

[32] Blake Ross, Collin Jackson, Nicholas Miyake, Dan Boneh, and John C. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security Symposium*, 2005.

[33] Andrew J. Saykin, Sterling C. Johnson, Laura A. Flashman, Thomas W. McAllister, Molly Sparling, Terrance M. Darcey, Chad H. Moritz, Stephen J. Guerin, John Weaver, and Alexander Mamourian. Functional differentiation of medial temporal and frontal regions involved in processing novel and familiar words: an fMRI study. *Brain*, 122(10):1963–1971, 1999.

[34] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *HotSec'10*.

[35] Norbert Schmitz. Improved guessing of composite passwords. Master's thesis, Ruhr University Bochum, March 2012.

[36] Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Blase Ur, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Correct horse battery staple: exploring the usability of system-assigned passphrases. In *SOUPS'12*, Washington, DC, USA, 2012.

[37] SkullSecurity. Leaked passwords (database). http://www.skullsecurity.org/wiki/index.php/Passwords.

[38] Sidney L. Smith. Authenticating users by word association. *Computers and Security*, 6(6):464–470, 1987.

[39] Techcrunch. 370 passwords you shouldn't (and can't) use on Twitter. Dec. 27, 2009. http://techcrunch.com/2009/12/27/twitter-banned-passwords/.

[40] telega and TechnalXS. Simple Mail: Mail client (POP3/IMAP/SMTP) for Firefox. https://addons.mozilla.org/en-us/firefox/addon/simple-mail/.

[41] Umut Topkara, Mikhail J. Atallah, and Mercan Topkara. Passwords decay, words endure: secure and re-usable multiple password mnemonics. In *ACM Symposium on Applied computing (SAC'07)*, Seoul, Korea, 2007.

[42] Percy Wegmann. jspos - Javascript part of speech tagger. Javascript port of Mark Watson's FastTag. https://code.google.com/p/jspos/.

[43] Charles Matthew Weir. *Using probabilistic techniques to aid in password cracking attacks*. PhD thesis, Florida State University, Tallahassee, FL, USA, March 2010.

[44] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *ACM CCS'10*.

[45] Nicholas Wright, Andrew S. Patrick, and Robert Biddle. Do you see your password? applying recognition to textual passwords. In *SOUPS'12*, Washington, DC, USA, 2012.

[46] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5), 2004.

[47] Bennet Yee, David Sehr, Greg Dardyk, Brad Chen, Robert Muth, Tavis Ormandy, Shiki Okasaka, Neha Narula, and Nicholas Fullagar. Native client: A sandbox for portable, untrusted x86 native code. In *IEEE Symposium on Security and Privacy*, May 2009.

[48] ZDNet. Microsoft: Write down your passwords. News article (May 23, 2005). http://www.zdnet.com/microsoft-write-down-your-passwords-1139193117/.

A Selective Regeneration and Brute-force Attacks

Selective regeneration of words. The user is allowed to selectively regenerate individual words in the passphrase. Each time the user discards a word they reduce the available choices and weaken the passphrase. We restrict the number of re-selections to the number of words in the phrase (e.g., the user can regenerate 6 words when constructing an 6 word passphrase) in order to ensure the user does not introduce too much predictability. For example, if a RS passphrase consists of six words from a 1024-word dictionary then the entropy will be: $log_2(1024^6) = 60$. If the user selectively regenerates six words, the entropy is reduced to: $log_2(1018^6) = 59.95$. Since only a few iterations are allowed, the entropy will not suffer significantly.

The selective regeneration feature will probably be more useful for the CD variant. Some selected words will not create coherent sentences. A user is more likely to reject these words,

which decreases the available combinations for that template. For example, assume the chosen template is: "noun verb adverb determiner noun verb preposition noun" with POS class sizes (noun, 1975), (verb, 1209), (adverb, 86), (determiner, 21), (preposition, 86). A randomly generated passphrase from this template, e.g., "discomfort rang down the sunlight wait of freedom" will provide an entropy of $log_2(1975 \times 1209 \times 86 \times 21 \times 1975 \times 1209 \times 86 \times 1975) = 70.57$ bits. Assume the user regenerates the following words: (discomfort→clamor), (the→a), (sunlight→heap→yelp→spirit) and (wait→answering→alter→contrived). Now the sentence becomes: "clamor rang down a spirit contrived of freedom" and the modified entropy is: $log_2(1971 \times 1206 \times 86 \times 20 \times 1971 \times 1206 \times 86 \times 1971) = 70.48$ bits. As appears from this example, after a few iterations, the sentence may converge to something acceptable to the user, without losing much entropy. However, if an attacker uses sophisticated natural language processing, they may be able to determine which words are more likely to be rejected, thereby narrowing their search space. A larger dictionary can help offset this loss of complexity.

Iteration count and brute-forcing Myphrase passwords. We experimented with different hash iteration counts, and found that 32768 caused an acceptable delay for our PC (Firefox addon) and smartphone (Android app) implementations. Running on a 2.5GHz Intel i7-2860 CPU, it required 2.84 ± 0.02 seconds to complete for the addon. On a 1.2GHz ARM Cortex-A8 HTC smartphone it required 3.23 ± 0.06 seconds. With assembler and hardware crypto accelerated instructions (e.g., OpenSSL) the time to iterate PBKDF2 reduced to 0.3 seconds on average. Note that, the extension adds almost 10 times more inefficiency in the PBKDF2 calculation, which benefits the attacker. Native code execution within the browser, e.g., Google Native Client [47], can be used to reduce this inefficiency. An attacker can also parallelize the computations to brute-force the dictionary. With a 60-bit passphrase, the attacker will need to perform 2^{59} calculations on average. On a 4-core PC this will require: $\frac{2^{59} \times 0.3}{60 \times 60 \times 24 \times 365.25 \times 4} = 1370020420$ years to search this space. If the attacker has access to a grid of 1 million such 4-core CPUs (e.g., a million-node botnet), it will still require 1370 years; without the hash iteration, the required time is only about 15 days (on average). For a 40-bit passphrase, the required times on the million-node grid are about 12 hours (with hash iteration) and 1.25 seconds (without hash iteration) on average. An attacker may be able to reduce the time using custom hardware (e.g., FPGAs). On the other hand, the attacker's workload to brute-force a target password will increase significantly if the server-side stores only a one-way mapping of the password obtained through the use of iterated hash/PBKDF2 functions with unique salt values per account.

B Memorability of Myphrase passphrases

Our hypothesis is that Myphrase passphrases may retain security advantages of random phrases without being too difficult to remember. We would like to test this hypothesis

through a user study in the future. Below we discuss some related work supporting the idea that personally meaningful words could be more memorable.

An fMRI study performed by Saykin et al. [33] shows that brain activity is much greater, and takes place in more regions, when a subject is shown a familiar word as compared to an unaccustomed word. Gregg [13] also suggests that recall of common words is higher than uncommon words. (Note that frequency ranked words in Myphrase are by definition "common" to the user.) Hulme et al. [17] performed two experiments: in the first, non-word sequences were found to be less memorable than words. The second experiment compared memorability of Italian and English words on English-speaking participants. Users remembered English words better, but memory span for Italian words increased after learning the English translations. This also supports the idea that a user will better remember words for which their semantics are well known.

The auto-suggest feature of Myphrase reduces typing and may also jog the user's memory (i.e., the task of recall is partially reduced to recognition). However, it is unclear to what extent this will help for overall memorization of the user's passphrase, including sequence. Early research by Gregg [13] suggests that recognition of words is actually higher for uncommon words. In a recent recognition study [45], in the context of text based passwords, it was found that recognition of words was no better than free-recall.

The connected discourse variant also leverages the semantic and syntactic structure of a sentence. Previous research (e.g., [25, 24]) suggests that such high-order patterns are more memorable than random sequences of words. A recent study on the memorability of random passphrases performed by Shay et al. [36] contradicts this notion, and indicates that there is little difference between recollection of random passwords versus passphrases. However, in this study (also in [45]) personally meaningful words were not used. Furthermore, users were not provided any recognition cues, and did not benefit from the repeated use of the phrase (one Myphrase passphrase is expected to be used repeatedly for all or most web logins).