Telepathwords: Preventing Weak Passwords by Reading Users' Minds

Saranga Komanduri, Richard Shay, and Lorrie Faith Cranor, *Carnegie Mellon University;*
Cormac Herley and Stuart Schechter, *Microsoft Research*

https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/komanduri

This paper is included in the Proceedings of the
23rd USENIX Security Symposium.

August 20–22, 2014 • San Diego, CA

ISBN 978-1-931971-15-7

Telepathwords: preventing weak passwords by reading users' minds

Saranga Komanduri, Richard Shay, Lorrie Faith Cranor
Carnegie Mellon University

Cormac Herley, Stuart Schechter
Microsoft Research

Abstract

To discourage the creation of predictable passwords, vulnerable to guessing attacks, we present *Telepathwords*. As a user creates a password, Telepathwords makes real-time predictions for the next character that user will type. While the concept is simple, making accurate predictions requires efficient algorithms to model users' behavior and to employ already-typed characters to predict subsequent ones. We first made the Telepathwords technology available to the public in late 2013 and have since served hundreds of thousands of user sessions.

We ran a human-subjects experiment to compare password policies that use Telepathwords to those that rely on composition rules, comparing participants' passwords using two different password-evaluation algorithms. We found that participants create far fewer weak passwords using the Telepathwords-based policies than policies based only on character composition. Participants using Telepathwords were also more likely to report that the password feedback was helpful.

1 Introduction

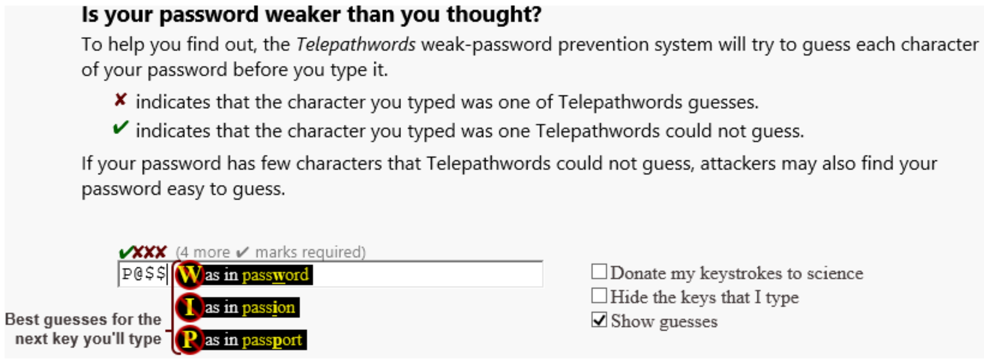
Users are often advised or required to choose passwords that comply with certain policies. Passwords must be at least eight characters long. They must contain characters from at least three out of four character categories (uppercase characters, lowercase characters, digits, and symbols). The password should not be based on a dictionary word.

While rules for composing passwords often feel arbitrary and capricious, they respond to a problem of genuine concern: left to their own devices, a significant fraction of users will choose common passwords that attackers may guess quickly. Composition rules were created decades ago under the assumption that minimum-length and character-set requirements would result in passwords that were harder for attackers to guess. It is only in the past few years that researchers have begun to test this hypothesis (and found the evidence to support it far weaker than assumed).

Indeed, password-composition rules feel arbitrary and capricious because, quite simply, they often are. Users can hardly be blamed if they question the credibility of rules that reward those who choose the common password `P@ssw0rd` over those who enter a long randomly generated string restricted to lowercase letters (e.g., to facilitate typing on a touch-screen keyboard) or of password meters that offer irreconcilably different quality estimates for the same string [4]. If we are to prevent users from selecting weak passwords, we must first improve the technology used to identify weak choices, but also overcome any skepticism caused the failure to clearly explain the need for the restrictions being imposed.

Our proposal, *Telepathwords*, is different from previous weak-password prevention schemes in that, as users enter their proposed password, it shows its best predictions for the next character they will type in real time (see Figure 1). Telepathwords makes these predictions using knowledge of common behaviors users exhibit when choosing passwords, common strings they frequently use to construct passwords, and a general model of the user's language. Telepathwords presents users who enter weak passwords with immediate and compelling evidence that their intended password may be easier to guess than they had previously assumed: a display of the characters they are about to type.

We describe the design, implementation, human-subjects testing, public deployment, and user response to the Telepathwords system. The results of our security testing are particularly compelling. In a 2,560-person Mechanical Turk study, passwords created using Telepathwords significantly outperformed (using both entropy and guessing number metrics) those created under length and character composition policies, while remaining as memorable as passwords chosen with the least stringent requirement (an eight-character


Is your password weaker than you thought?

To help you find out, the *Telepathwords* weak-password prevention system will try to guess each character of your password before you type it.

✗ indicates that the character you typed was one of Telepathwords guesses.

✓ indicates that the character you typed was one Telepathwords could not guess.

If your password has few characters that Telepathwords could not guess, attackers may also find your password easy to guess.

✓✗✗✗ (4 more ✓ marks required)

P@$$ | W as in password | I as in passion | P as in passport

Best guesses for the next key you'll type

☐ Donate my keystrokes to science
☐ Hide the keys that I type
☑ Show guesses

Figure 1: The Telepathwords system, shown here as deployed in a publicly available password-weakness checker, attempts to guess the next character of a users' password *before* he or she types it.

minimum-length requirement). They matched or slightly beat passwords created under a policy that checked against a large cracking-dictionary, while at the same time having more users state that they found the visual feedback useful. The improvements in guessing resistance were most pronounced for the most vulnerable part of the distribution. That is, the weakest passwords created using Telepathwords require orders of magnitude more guesses than the weakest passwords created under policies based on composition and length. This suggests that Telepathwords can offer meaningful improvement in defending against online guessing attacks; an improvement that we hope can rebuild users' confidence that the constraints being imposed on them are indeed necessary.

2 Design and Implementation

We begin our discussion of the Telepathwords system by describing the intended user experience, then discuss the overall architecture and prediction algorithms required to implement that experience. We also describe the feedback mechanisms we included to observe usage of the system, as well as the limitations inherent to our implementation.

2.1 User Experience

Telepathwords enhances the text field into which users type new passwords with two additional elements: a *prediction display* and a *feedback bar*. Figure 1 illustrates both, with the prediction display just to the right of the typed password (`P@$$`) and the feedback bar immediately above it.

2.1.1 Prediction display

The prediction display shows the three characters (or fewer) that Telepathwords predicts the user is most likely to type next. As users are most likely to be familiar with prediction from autocomplete, where the predictions represent a *desirable* mechanism to save labor, we needed to emphasize that the characters telepathwords predict are *undesirable*, as these choices are least likely to make the password harder for attackers to guess. We thus display predicted characters in block uppercase within the prohibition symbol, or 'universal no symbol': a red circle with a slash through it. We anticipated the symbol would be familiar to users because it is standardized (ISO 3864-1, though we did not strive to achieve full compliance in our use), widely used in road signs, and pervasive in popular culture such as t-shirts and movies.

To the right of the character we present a short explanation of why that character was predicted. If we predicted the character because we detected the user typing a repeating sequence of characters, we display 'repeating' followed by the character sequence being repeated. If it is the next character of a common string, we present the words 'as in' followed by that string, with the next character boldfaced and underlined. For example, in Figure 1, the input of "P@$$" yields predictions: "W as in pass**w**ord," "I as in pass**i**on," and "P as in pass**p**ort."

2.1.2 Feedback bar

In the feedback bar above the password-entry field, we show either a checkmark or crossout symbol aligned directly above each of the characters already typed. A checkmark means the character was not predicted by Telepathwords, whereas a crossout indicates it was one of the characters guessed. We also display a crossout if the user types a common substitute for one of the predicted characters, such as an `@` to avoid using an `a`. To the right of these symbols we provide guidance as to how many more hard-to-guess characters are recommended, or would be required if Telepathwords were deployed with a particular minimum hard-to-guess character requirement. For example, Figure 1 shows one check and three crossouts above the user input "P@$$" since each of the last three characters was predicted based on the characters that came before it.

2.1.3 Special cases

In many applications, password-creation fields are configured to hide the keys typed, replacing them with a generic symbol (usually a solid circle or an asterisk). When the password field is configured to hide the characters that have been typed, we also replace those characters with a solid circle in our prediction string. The predicted characters are still shown.

When users type a common substitute for a predicted character, such as a *$* when an *s* is predicted, we display the following message customized for the replacement:

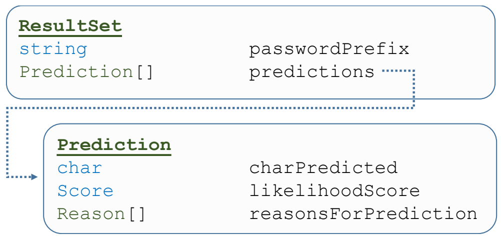
> Replacing a predictable letter with a key that looks similar?
>
> Attackers also know to substitute `$` for `s`, so it does little to improve your password.

We faced a particularly delicate conundrum in how to handle predictions that completed profanities. An examination of the Rockyou leaked dataset reveals that profanities are not uncommon choices. Unlike applications of prediction in search queries, we could not simply remove these predictions, as this would lead users to believe falsely that profane passwords were less weak than they actually are. On the other hand, we could not display profanities to users who might have no intent of typing them, and who might be minors. We decided that providing good security advice mandated that we predict the next character, but we replace the rest of the profane string with a string of solid circles in the explanation of the prediction. We also display a pop-up message if users complete a profanity, alerting them to the fact that profanities are common in passwords and thus quite predictable. In crafting this message, we decided to embrace the inevitability that some users might find humor in our attempts to hide profanity.

> Do you email your mother with that keyboard?
>
> Many people include profanity in their passwords. Attackers know this. If you also use profanity, you'll just make your password easier for attackers to guess.

2.2 Architecture

Telepathwords employs a client-server architecture, using JavaScript to present a front-end user interface using predictions asynchronously queried from a prediction server. The constraints of client-side prediction would not have allowed our prediction engine to use a 1.5GB language corpus (see Section 2.3.1), which we hope to grow in order to increase prediction quality and recognize additional languages.



```
ResultSet
string              passwordPrefix
Prediction[]        predictions

Prediction
char                charPredicted
Score               likelihoodScore
Reason[]            reasonsForPrediction
```

Figure 2: When the client queries the server with a password prefix, the Telepathwords prediction engine generates a result set containing a series of predictions, each of which may have been predicted based on a number of reasons (e.g., a dictionary match or a keyboard pattern).

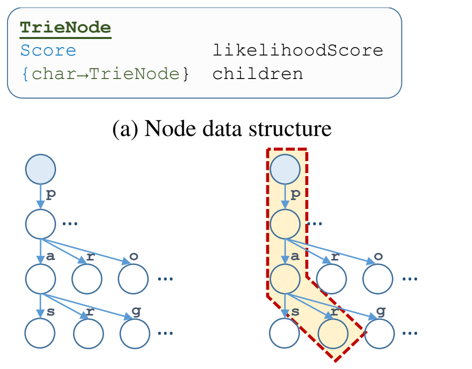
Other weak-password-prevention systems, such as common password meters, eschew server-side predictions. One justification is security. However, the current architecture of the web necessitates that whatever password the user eventually chooses will inevitably be sent to the website's servers in a plaintext-decryptable format. To prevent the size of a prediction from revealing the prefix sent to the server, we use a custom format to compress and then pad responses to a common length. We route all client-server communications over HTTPS.

A second reason to eschew server-side predictions is performance. However, network latencies are relatively small in comparison to users' expectations of response time, and can be made smaller by moving servers closer to users and pre-fetching likely queries, as demonstrated by the speed of auto-complete in web search. For example, though our deployment used servers in a single geographic location to serve users worldwide, the median latency between key-up and the rendering of a prediction at the client was a fifth of a second (see Section 3).

One additional security risk we decided to take was to maintain a cache of previously queried prefixes on the server, whereas we would otherwise be able to delete all evidence of a past request after serving a prediction. This greatly increases the likelihood that when the n^{th} character of a password arrives, the server will already have done the work to process the first $n-1$ characters.

2.3 Prediction algorithms

When performing a prediction, we create a result set data structure and populate it with a set of predictions, as illustrated in Figure 2. Each prediction object represents a possible next character of the password and a score that indicates its estimated likelihood. There may be more than one reason to predict a character, and so each prediction object contains a set of reason objects. We populate the result set by spawning a set of *predictors*, algorithms which identify reasons for predicting a character



```
TrieNode
Score                likelihoodScore
{char→TrieNode}      children
```

(a) Node data structure

(b) A section of the trie (c) Descent to node *par*.

Figure 3: The trie data structure maps strings to likelihood scores. Nodes (circles) with higher scores appear to the left of lower-scoring siblings. Subfigure (c) illustrates a walk to the node storing the likelihood score for string *par*.

will be typed next, add that reason to the prediction object for that character, and increase the predictions score as necessary.

When all the predictors have run, we rank the predictions and reasons. Before sending predictions to the client, we discard predictions and reasons that are not ranked high enough to be displayed to the user. We cache the result set so that we can use it again for future queries for this string, or extensions of this string.

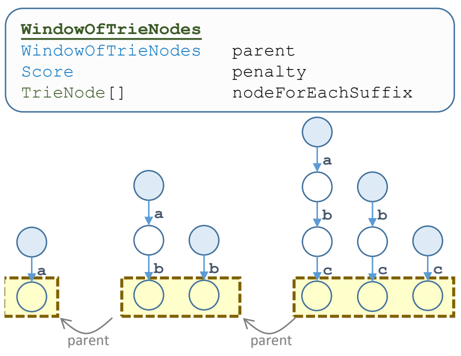
Telepathwords currently contains predictors for common character sequences, keyboard movements, repeated strings, and interleaved strings.

2.3.1 Common character sequences

This predictor detects known prefixes of common character sequences from language models and databases of common passwords, and predicts the remaining suffix. The expected likelihood of the prediction increases with the length and frequency with which the prefix was observed when the model was built.

To search quickly through a large prefix of known strings and their frequencies, we use the space-efficient completion trie of Hsu and Ottaviano [10], as illustrated in Figure 3. The trie used by Telepathwords contains a 1.5GB English-language model derived from browser search queries and a set of passwords that occurred five times or more in the RockYou dataset. We removed all capitalization and spaces from the language model before building the trie.

Completion tries are already used for auto-completion and word-breaking applications, and these applications require algorithms that adapt to common misspellings and typos. For example, existing systems will walk a



```
WindowOfTrieNodes
WindowOfTrieNodes     parent
Score                 penalty
TrieNode[]            nodeForEachSuffix
```

Figure 4: Telepathwords uses a sliding window to walk the trie for each suffix of the queried string. If the query string is a single character (e.g., *a*), the sliding window will contain only one node (left). A two-character string (*ab*) will a sliding window that walks the trie to two different nodes (center). The query string *abc* yields a window that covers the suffixes *c*, *bc*, and the full suffix *abc* (right). Adding one character to the query causes the pointer to each node to descend to the child node for that added character, and creates a new node in the window by stepping from the root node to the added character. Telepathwords may add a penalty to the window when the path down the trie is different from the actual string queried, such as if a window is created to represent a transposition.

completion trie reversing the two characters at the suffix, applying a penalty to account for the fact that transpositions occur with much lower frequency than correctly sequenced characters. If the transposed prefix occurs with sufficient frequency to overcome the penalty, the system may continue to track that transposition and make predictions based on it.

Since Telepathwords uses tries to look for common strings that may begin anywhere in the query (e.g., `passw` in the query `notapassword`), we maintain a window of completion-trie nodes for each possible starting position, as illustrated in Figure 4. We track the trie node for each possible suffix of the query. In addition, we maintain two special windows: one that walks the trie only when letters are typed and one that does so only when digits are typed. These special windows help to detect words broken up by non-alphabetic characters (e.g., `pa1234ssword`) or numbers broken up by non-digits (e.g., `12x34y678z9`).

In contrast to other applications of tries, users choose passwords with the deliberate goal of creating a string that is hard to predict, leaving many more anomalies to detect and work around than if divergences from known strings occurred only by accident. We thus maintained a large list of windows for each queried password prefix so as to preserve nodes that might not immediately appear

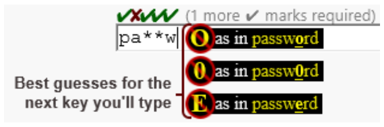

(1 more ✔ marks required)
pa**w
O as in password
0 as in passw0rd
E as in passwerd
Best guesses for the next key you'll type

Figure 5: The **common character sequence** predictor walks the ancestry chain to see if a completion that was broken by an unpredicted character might still provide the best guess for what happens next. For `pa**w`, the third ancestor (`pa`) predicted the `w` in the fifth position, and since there are no other likely predictions, predictions using this ancestor reach the top.

valuable but might prove predictive as more characters arrive.

We also built a table mapping common character substitutions, such as `3` for `e`, `$` for `s`, and `0` for `o`, that are often provided in password-creation guidance (in our view, misguidedly). If we detect a character that is often substitute for another, we create a window using the character we believe was substituted for and assign that window an appropriate penalty.

To detect when users type distractor characters in place of predicted characters, then carry on with the predicted string, we walk up the ancestry path of the current prefix to look for predictions that may have been abandoned due to such behavior. For example, if the user has typed the prefix `pa**w`, the algorithm will walk up from `pa**w` to the ancestor prefix `pa`, determine that the prediction of `password` for this prefix would have correctly predicted the `w` in the fifth position, and may thus revive that prediction to predict a `o` in the next position. See Figure 5. Similarly, we use the standard error-correction technique of detecting when a user has skipped a key and typed the second character predicted in place of the next character predicted.

The analysis of each password prefix of length n begins with the analysis of its immediate prefix of length $n-1$. Thus, the cost of analysis grows at least linearly with the length of the password. We maintain a main-memory cache of recently analyzed query strings so that results can be re-used when the suffixes of a previously-queried string are queried.

Even under heavy load, the cache is small in comparison to the 1.5GB language corpus. In our deployment, the language corpus is stored in main memory. During development, we found performance to be sufficiently fast using a solid state drive (SSD) to store the corpus and only mapping pages into main memory on demand. In our deployment, we prefetched the full corpus into DRAM as our servers did not have SSDs.

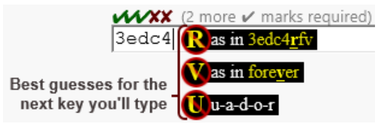

(2 more ✔ marks required)
3edc4
R as in 3edc4rfv
V as in forever
U u-a-d-o-r
Best guesses for the next key you'll type

Figure 6: The password `3edc4`, composed of vertical columns on a QWERTY keyboard (`3edc`, `4rfv`, etc.), triggers the **keyboard-movement predictor** yielding `r` as the top guess for the next character. The second prediction guesses that the `4` is used in place of `for` in `forever`, and the third prediction guesses that `ecuador` is interleaved into every other character of the password.

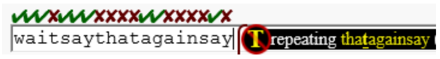


Figure 7: The start of a repeating string triggers the **repetition predictor**.

2.3.2 Keyboard movements

We developed this predictor to detect passwords composed of a sequence of characters typed by moving one's finger over a sequence of adjacent keys.

We built a keyboard model that maps characters to x and y coordinates that represent the column and row of the key used to type each character on a keyboard. We represent an n-character password prefix as a sequence of n key positions, then generate a series of $n-1$ movements from the first to the last character. We then work backward from the end of the prefix to count the number of consecutive moves that are to adjacent keys and, of those, the number of consecutive moves in the same direction. We count movements that wrap from one end of the keyboard (e.g., from top to bottom) as adjacent.

We have currently mapped only QWERTY keyboards, but the implementation is generalized to support any mapping of characters to coordinates.

2.3.3 Repeated strings

This predictor looks for instances of repeated strings in password prefixes. For each possible suffix of length n, it looks for repeated sequences of the suffix. The longer the repeated sequence, the stronger the prediction. If the repetitions are adjacent to each other (`xyabcabcabc`), then the predictor guesses the next character in the repeated sequence (or the first if the end has been reached). If the suffix and its copy are not adjacent, then the early copy and the intervening string are assumed to be in the process of repeating. For example, in `abcdefabc` the suffix `abc` is repeated twice and the predictor guesses that `def` will come next.

2.3.4 Interleaved strings

This predictor looks for passwords composed of two predictable strings interleaved with each other, such as `p*a*s*s*w*o*r*d` or `ppaasswwoorrdd`. It splits passwords to separate the odd- and even-indexed characters and runs the other predictors (with interleaving-detection turned off) on the substrings. If, for example, the next character is at an even-index, it uses the even-index substring to make the prediction, and also examines the predictability of the odd-index substring in evaluating the likelihood that the query actually represents two interleaved strings.

2.4 Telemetry

Our public deployment of Telepathwords maintains a limited log of user behaviors, including page loading, resizing, key-up events, and prediction rendering events. Unless users explicitly opt-in to 'donate' their keystrokes to science, we record the timing of keyup events, the number of keys added or deleted, and the position of the change, but not the actual keys typed. We also record whether characters currently in the password field were among those predicted, recording data similar to that which is displayed in the feedback bar.

While we store logs online, the server is unable to read their contents. At the start of a user session the client-side JavaScript requests a one-time session-encryption key from the server. The server generates the key, encrypts it with a public key, and then writes the encrypted session key to the first entry of the log for the session. It then sends the key to the client and maintains no further record of it. The private key is not stored on any publicly facing server. The client XORs the log data stream with a bit stream generated by using AES in counter mode with the Stanford Javascript Crypto Library (SJCL) [27]. We opted for this approach, inspired by Kelsey and Schneier [23], because of its simplicity and as concerns over confidentiality far outweighed that of integrity. As logs are never read online, and no action is taken with them but to store them, we do not know of a scenario in which an adversary could learn the contents of the logs by modifying them.

2.5 System Limitations

The current deployment of Telepathwords has some limitations that are inherent to research prototypes. The language corpus is US-centric and somewhat dated, and so unlikely to pick up on words or phrases uncommon in the United States or that have entered the common lexicon since 2012. An ideal set of corpora would be international and receive constant updates from the latest search queries, news, and other topical sources.

Telepathwords cannot currently detect reversed character sequences (`gfedcba` in place of `abcdefg`) unless that reversal is itself already common enough to be in the language corpus (as it is for `drowssap`, for example). One way to implement reversal detection would be to reverse the more common strings in our language corpus, assess a penalty for the reversal, and insert them into our completion trie.

The privacy promises made by the current deployment of Telepathwords prohibit analysis of passwords for any purpose other than the issuing of predictions, and so the language corpus, scoring rules, and known set of common password-creation behaviors do not grow over time. Thus, if users flock to common behaviors in response to Telepathwords (as they do in response to password-composition rules) we may not be able to detect these behaviors in the current deployment.

3 Deployment

Our first deployment of the Telepathwords technology is a password-testing website, similar in purpose to existing websites that offer to test the 'strength' of passwords [9, 16, 20], which is hosted at https://telepathwords.research.microsoft.com. We took great pains to avoid positioning the service as measuring any form of 'strength' or 'security', as no system can be certain that any user-chosen password is truly strong or secure. There is no guarantee that a password that *appears* strong would not be predictable by an attacker with better knowledge of how certain users construct passwords.

As with any publicly facing Internet service, we deployed Telepathwords with some trepidation not knowing what usage levels to expect and not knowing what factors we may have failed to anticipate when performing load-testing experiments. In our pre-deployment throughput tests, Telepathwords processed 454,486 passwords in a database of breached Yahoo! Voices passwords in under 7 hours using 3 cores of a 3Ghz Xeon E5 1607 (roughly five passwords per core-second.)

We opened up our system to the public on December 5, 2013 and saw our highest usage rates shortly afterward, as the technical press published articles about the release.

3.1 Data collected

We downloaded our encrypted logs to a researcher's workstation for decryption and analysis. We graph the arrival rate of users to our service in Figure 8, which illustrates the burst of traffic during initial release dissipating over time. We are also able to observe the delay experienced by users between the time they typed a key and received a prediction for what the following key

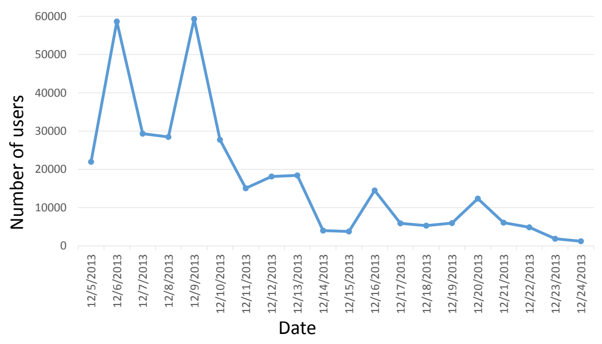


Figure 8: Sessions served per day by the Telepathwords service shortly after release. (A Session is counted when the Telepathwords page loads and the server receives a request for a session ID and encryption key used for logging.)

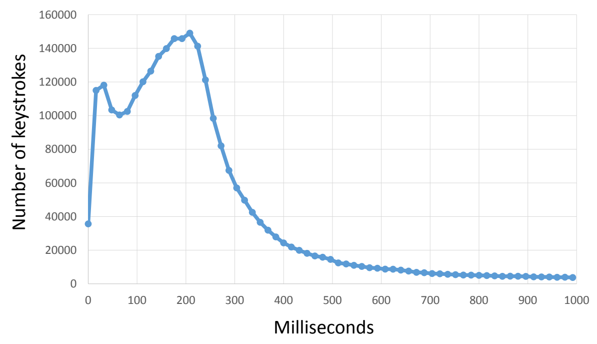


Figure 9: The distribution of the delay between the "keyup" and "render" events for all keystrokes during the recording period. The median occurs at 208ms.

would be, graphed in Figure 9. The median delay was 200ms. A peak in the graph around 20ms is likely due to fast rendering of predictions cached within the browser.

We are also able to use the logs to track how much activity users perform during each user session. In Figure 10, we examine the distribution of number of keys pressed per user session, seeing that some users appeared to use the site to test multiple passwords.

4 Experimental Methodology

In addition to the deployment, we conducted a comparative evaluation of Telepathwords and a number of existing password-composition policies via a two-part online study using Amazon's Mechanical Turk crowdsourcing service. To facilitate comparisons with prior work, much of our methdology mirrors that of a recent line of research from Carnegie Mellon University, including that of Kelley [12], Komanduri [13], Mazurek [15], Shay [24, 25], Ur [28], and others.

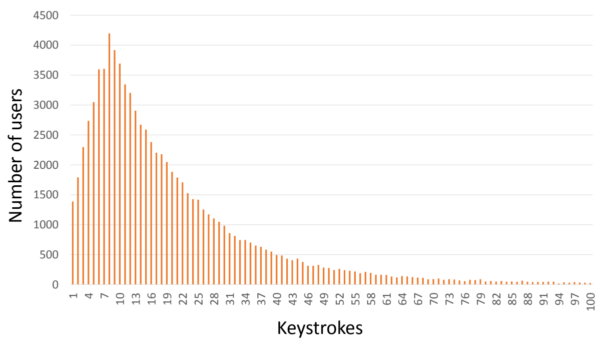


Figure 10: The number of keystrokes received per user session provides insight into user engagement with the site. The median is 15 and the mean is 21 keys pressed per session.

Our experiment was approved by Carnegie Mellon University's institutional review board prior to the start of our study.

4.1 Recruiting and Data Collection

We recruited participants from Amazon's Mechanical Turk by listing a Human Intelligence Task (HIT) in which we offered 55 cents to "Take a 5-minute survey with 70-cent bonus opportunity!" We required participants be 18 years of age and located in the United States.

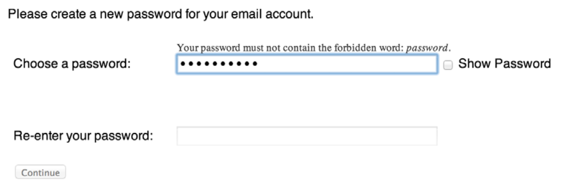
We asked participants to imagine that their email account had been compromised and that they needed to create a new password to replace it. We used a round-robin algorithm to assign participants one of six password-composition policies. As users typed their proposed password, we provided real-time feedback indicating the conditions that needed to be met for participants to satisfy their assigned policy. Whereas prior CMU studies checked compliance with password policies after participants had submitted them, in this study we enabled the submit button only after a participant had satisfied the policy (and correctly retyped the password).

After participants submitted the password, we presented a survey with up to 24 questions to ask about their experience creating the password, more general questions about their password habits, and their demographics. Following the survey we asked participants to recall their passwords, giving them five attempts to do so. We displayed their password to them if they could not recall it within those five attempts. This concluded *part one* of our study.

Two days later, we invited participants to return for *part two* of our study, sending them an email via an interface provided by Mechancial Turk. We offered 70 cents to return for this HIT, in which we asked participants to recall their passwords. Again, we allowed participants five attempts to provide the correct password. We displayed participants' passwords if they were unable to succeed within five attempts, though we did not tell them this a priori. We wanted participants to complete the study whether or not they recalled their password, so we provided them with a last-resort mechanism for recovering their passwords: a link, which would send an email, which contained a link, which led to a webpage, which displayed the correct password. We took this intentionally circuitous approach, rather than simply showing participants their passwords on request, to discourage them from using the recovery mechanism without first trying to recall their passwords. Outside the extra effort for password recovery, we did not further penalize participants for failing to recall their passwords; if we had, and future participants learned about it, they might have been more likely to store their passwords.

Finally, we asked participants to take an 18-question survey asking about their password-recall process and whether they had stored their passwords.

Except as noted, we focus our analysis on those participants who finished the first part of our study. Our analysis of dropout rates examines all participants who begin the study, and our analysis of part two examines only those participants who finished part two. We exclude participants from part two if they did not complete it within three days of the invitation.



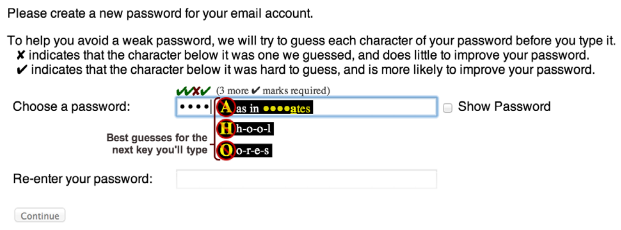
Please create a new password for your email account.

Your password must not contain the forbidden word: *password*.

Choose a password: ••••••••••• Show Password

Re-enter your password:

Figure 11: The 3class8-d treatment on the experimental website.



Please create a new password for your email account.

To help you avoid a weak password, we will try to guess each character of your password before you type it.
✗ indicates that the character below it was one we guessed, and does little to improve your password.
✓ indicates that the character below it was hard to guess, and is more likely to improve your password.

✓✓✗✓ (3 more ✓ marks required)

Choose a password: •••• Show Password

Best guesses for the next key you'll type

A as in ●●●●ates

H h-o-o-l

O o-r-e-s

Re-enter your password:

Figure 12: The telepath treatment on the experimental website.

4.2 Treatments

The only features of our study that varied between participants were the assigned password-composition policy, whether the password field hid the characters typed into it, and a few survey questions about policies specific to certain treatments. Of the six password policies we assigned to participants, two use a Telepathwords-based policy and four use policies based on composition-rules and (in one case) dictionary checks.

- **telepath, telepath-v** These two conditions employed a Telepathwords-based policy that required users to provide a password with at least six characters that were not predicted by the system. The system does not predict the first character, and so the first character of each password always counted toward the requirement. The two conditions differed only in that passwords would be shown by default as they were being typed in telepath-v and were hidden by default in telepath.
- **basic8** This condition required passwords of at least eight characters in length.
- **3class8** This condition also required passwords of at least eight characters in length, adding the requirement that the password include three of four character classes: uppercase letters, lowercase letters, digits, and symbols. This policy mirrors the default password policy for Microsoft Windows Active Directory.
- **3class12** This condition required passwords of at least 12 characters in length from three of four character classes.
- **3class8-d** This condition required passwords to include at least eight characters, from three of four character classes, and required that the string of all letters within the password not match any of the roughly 3M words in the free Openwall cracking dictionary [5].

We displayed the requirements that had not yet been met directly above the password-entry field, as shown in Figure 11. If the password had not yet met the length requirement, we displayed that requirement. If a password met the length requirement, we displayed remaining composition requirements, if any. If the password met the length and composition requirements but failed a dictionary check (for 3class8-d), we displayed the match and indicated that the password must not contain the matched word.

We displayed a checkbox that allowed participants to show or hide the characters being typed. With the exception of telepath-v, the password was hidden by default.

	basic8	3class8	3class12	3class8-d	telepath	telepath-v
Participation						
arrived at part one	476	475	472	469	476	476
finished part one	431/476 (91%)	440/475 (93%)	425/472 (90%)	402/469 (86%)	420/476 (88%)	442/476 (93%)
returned & finished part two in <3 days	270/431 (63%)	296/440 (67%)	277/425 (65%)	260/402 (65%)	267/420 (64%)	257/442 (58%)
Password Selection & Handling among part-two participants						
did not store	172/270 (64%)	197/296 (67%)	168/277 (61%)	155/260 (60%)	168/267 (63%)	141/257 (55%)
did not re-use	221/270 (82%)	228/296 (77%)	226/277 (82%)	214/260 (82%)	229/267 (86%)	203/257 (79%)
did not store or re-use	135/270 (50%)	149/296 (50%)	140/277 (51%)	118/260 (45%)	138/267 (52%)	112/257 (44%)
Password Recall in 5 tries without reminder						
during part one	423/431 (98%)	434/440 (99%)	414/425 (97%)	391/402 (97%)	407/420 (97%)	429/442 (97%)
all part-two participants	176/270 (65%)	213/296 (72%)	186/277 (67%)	193/260 (74%)	183/267 (69%)	178/257 (69%)
part two did not store	105/172 (61%)	131/197 (66%)	104/168 (62%)	103/155 (66%)	103/168 (61%)	86/141 (61%)
part two did not re-use	144/221 (65%)	163/228 (71%)	151/226 (67%)	155/214 (72%)	159/229 (69%)	143/203 (70%)
part two did not store or re-use	83/135 (61%)	97/149 (65%)	86/140 (61%)	73/118 (62%)	84/138 (61%)	69/112 (62%)

Table 1: We tally the set of participants who began part one of our study, finished it, and who returned for part two. We measure recall rates for part one (shortly after password selection) and part two. We break down part-two recall rates to factor out participants who reported re-using passwords they already knew or storing their passwords.

5 Experimental Results and Analysis

The application of multiple statistical tests increases the chance of producing a Type I error, finding a significant difference where none exists. To compensate for this, we use a standard two-step process. First, we only perform pairwise tests if an omnibus test is significant. We use the Kruskal-Wallis omnibus test (KW) for quantitative data and the χ^2 test for categorical data. Second, we correct all pairwise tests using the Holm-Bonferroni method (HC). We use the Mann-Whitney U for quantitative pairwise comparisons and Fisher's Exact Test and the χ^2 test for categorical pairwise comparisons.

We performed our experiment in February 2014. We recruited 2,844 workers to accept our HIT for part one of our study. Of these, 2,560 finished, received payment, and received invitations to return two days later. A total of 1,627 participants (64%) returned for the second HIT within three days of when we sent their invitation. Participants' demographics reflected a typical population of workers on Mechanical Turk, with a median reported age of 27, nearly 60% reporting as male, and 44% reporting having at least a bachelor's degree.

We show the progress of participants through our study in Table 1. We removed from our analysis five participants who created more than one password by using the back button or reloading the password-creation page.

The condition with the highest dropout rate was 3class8-d, while the lowest dropout rate was telepath-v. Table 2 shows the dropout rates for each condition. It also gives the test's p-value for the null hypothesis (that the difference between dropout rates was unaffected by condition) for each pair of conditions. For example, at $p = 0.01$ (resp. $p = 0.007$) the difference in dropout rates between 3class8-d and 3class8 (resp. telepath-v) is significant. For all of the other condition pairs the hypothesis that condition had no effect on dropout rate is not ruled out. Note that the table has a triangular structure since we list the result of each pairwise test only once, and this same format is used for our other categorical tests (i.e. Tables 3, 4, 5, 6 and 7).

			Fisher's Exact Test p (Holm-Bonferroni corrected)				
Treatment	Part one dropout		telepath	3class12	basic8	3class8	telepath-v
3class8-d	67/469	14%	1.000	.458	.318	.010	.007
telepath	56/476	12%		1.000	1.000	.318	.255
3class12	47/472	10%			1.000	1.000	1.000
basic8	45/476	9%				1.000	1.000
3class8	35/475	7%					1.000
telepath-v	34/476	7%					

Omnibus χ^2_5=19.373, p=0.002

Table 2: The fraction of participants who dropped out during part one, with corrected pairwise comparisons of all treatment groups.

The median time spent to create a password was 32 seconds for participants in basic8, 43 for 3class8, 53 for both 3class12 and 3class8-d, 85 for telepath-v, and 96 for telepath. We anticipated participants using Telepathwords might spend more time, as these treatments included three lines of instructions not present in other treatments (see Figure 12) and their novelty may have led to more exploration.

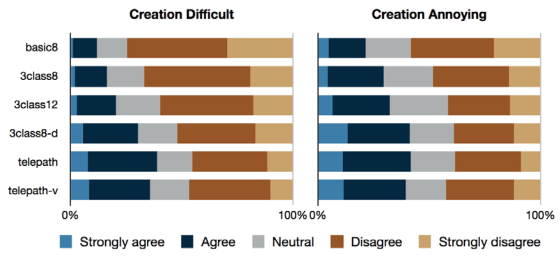


Figure 13: "Creating my password was difficult" and "Creating my password was annoying."

5.1 Recall

We provide recall rates for part one of the study for reference only, as just minutes had passed since participants had chosen their passwords. In the second section of Table 1, we see that 61.5% of participants indicated that they had not stored their password and that we had not detected them pasting or auto-filling a password into the recall field. Differences between treatment groups were not statistically significant (χ^2_5=9.231, p=0.1).

We looked at the number of part two recall attempts by the subset of participants who did not store their passwords, did not use the reminder feature, and did not re-use a previous password. Of these 502 participants, 79.3% entered the password on the first attempt, and 14.5% entered it on the second attempt. While the omnibus test shows a significant difference between conditions for taking more than one attempt (χ^2_5=13.943, p =0.016), the pairwise tests showed no significant differences. Among the 398 of these participants who entered their password correctly on the first attempt, the median password-entry time was 14.8 seconds; this did not vary significantly by condition (KW χ^2_5=4.705, p =0.453). Looking at the 1159 participants who did not use the reminder, 80.9% entered the password correctly on the first try. This differed by condition (χ^2_5=12.604, p =0.027), but no pairwise test was significant.

5.2 Participant Sentiment

In addition to recording participant behavior, we asked participants about their experience. We asked all participants whether they felt that creating their password was difficult or annoying, with results in Figure 13. We show the pairwise comparisons across conditions for difficulty in Table 3 and annoyance in Table 4.

The three policies that tested participants' passwords against lists of common passwords (the Telepathwords conditions and 3class8-d) had a greater proportion of participants who were annoyed than those using the purely composition-based policies. The differences with the simplest policies were significant, as shown in Table 4.

			Fisher's Exact Test p (Holm-Bonferroni corrected)				
Treatment	Creation difficult		telepath-v	3class8-d	3class12	3class8	basic8
telepath	163/420	39%	.374	.078	<.001	<.001	<.001
telepath-v	158/442	36%		.374	<.001	<.001	<.001
3class8-d	123/402	31%			.006	<.001	<.001
3class12	87/425	20%				.374	.005
3class8	73/440	17%					.209
basic8	51/431	12%					

Omnibus χ^2_5=135.199, p<.001

Table 3: The fraction of participants in each treatment who agreed that it was difficult to create a password during the experiment.

			Fisher's Exact Test p (Holm-Bonferroni corrected)				
Treatment	Creation annoying		3class8-d	telepath-v	3class12	3class8	basic8
telepath	175/420	42%	1.000	1.000	.034	.002	<.001
3class8-d	165/402	41%		1.000	.052	.004	<.001
telepath-v	175/442	40%			.117	.013	<.001
3class12	136/425	32%				1.000	.005
3class8	129/440	29%					.052
basic8	92/431	21%					

Omnibus χ^2_5=61.805, p<.001

Table 4: The fraction of participants in each treatment who agreed that it was annoying to create the password during the experiment.

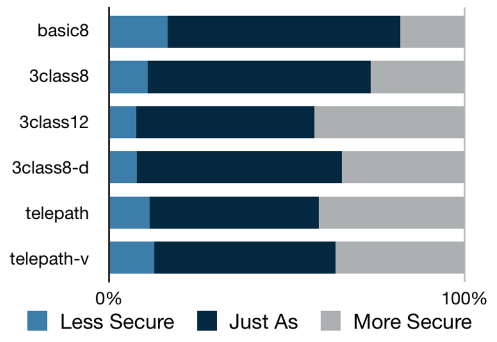


Figure 14: "When compared to the password I use for my primary email account, the password I created for this study was:".

We also asked participants whether they believed their study-created password to be more, less, or just as secure as their primary email password. The results are in Figure 14. Belief that the study passwords were more secure

		Fisher's Exact Test *p* (Holm-Bonferroni corrected)				
Treatment	More Secure	telepath	telepath-v	3class8-d	3class8	basic8
3class12	180/425 42%	1.000	.329	.134	<.001	<.001
telepath	172/420 41%		.552	.309	<.001	<.001
telepath-v	161/442 36%			1.000	.013	<.001
3class8-d	139/402 35%				.075	<.001
3class8	116/440 26%					.027
basic8	78/431 18%					

Omnibus χ^2_5=83.62, p<.001

Table 5: The fraction of participants who selected "More secure" in response to "When compared to the password I use for my primary email account, the password I created for this study was:".

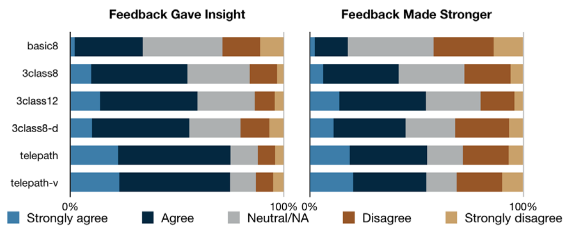


Figure 15: "The visual feedback I received gave me insight into the quality of my password" and "The visual feedback that was displayed helped me to create a stronger password that I would have otherwise."

		Fisher's Exact Test *p* (Holm-Bonferroni corrected)				
Treatment	Gave Insight	telepath-v	3class12	3class8-d	3class8	basic8
telepath	315/420 75%	1.000	<.001	<.001	<.001	<.001
telepath-v	331/442 75%		<.001	<.001	<.001	<.001
3class12	253/425 60%			.873	.678	<.001
3class8-d	224/402 56%				1.000	<.001
3class8	241/440 55%					<.001
basic8	146/431 34%					

Omnibus χ^2_5=208.104, p<.001

Table 6: Agreement with "The visual feedback I received gave me insight into the quality of my password."

		Fisher's Exact Test *p* (Holm-Bonferroni corrected)				
Treatment	Feedback Helped	telepath-v	3class12	3class8-d	3class8	basic8
telepath	231/420 55%	1.000	1.000	.029	.001	<.001
telepath-v	241/442 55%		1.000	.029	.001	<.001
3class12	231/425 54%			.033	.001	<.001
3class8-d	180/402 45%				1.000	<.001
3class8	183/440 42%					<.001
basic8	77/431 18%					

Omnibus χ^2_5=178.62, p<.001

Table 7: Agreement with "The visual feedback that was displayed helped me to create a stronger password that I would have otherwise."

ranged from 18.1% for basic8 to 42.4% for 3class12, and significant differences are in Table 5.

We displayed visual feedback in *all* conditions to help participants comply with their assigned policy. We asked participants if they believed the feedback gave them insight into their passwords and if it helped them to create better passwords. We show their responses in Figure 15 paired with significance tests in Tables 6 and 7.

The Telepathwords treatments, along with 3class12, had the greatest proportion of participants who believed the feedback helped them create a stronger password. A significantly larger portion of participants in the two Telepathwords conditions agreed that the feedback provided more insight than the other treatments—including the dictionary-based feedback in 3class8-d. This is tempered of course by the higher number who found password creation difficult or annoying with the tool. We see this as a hopeful sign that Telepathwords can help improve the credibility of technology designed to prevent users from choosing weak passwords.

5.3 Security Results

In Table 8 we present statistics summarizing the composition of passwords created under each policy, and security scores calculated by three metrics. We focus our analysis on the passwords identified to be weakest as an attacker is most likely to try these first. Dictionary attacks to obtain beachheads into organizations succeed when the first account is breached. Thus, improving the security of the weakest password in an organization by a small amount is far more likely to prevent an attacker from obtaining a beachhead than a large improvement to the average password would. This is particularly true for an online attack where a limited number of guesses per account can be tried.

We did not encounter any repeat passwords in our sample, so we cannot use frequency as a metric. Rather, the first metric we apply is an entropy calculation generated by the open-source `zxcvbn` password meter [30]. Its advantages are that it is publicly available, open-source, and already relied on by large-scale systems, including DropBox. Its primary disadvantage is that it was designed to meet the constraints required for deployment as a client-side password meter; it needed to be small enough to download quickly and efficient enough to run in JavaScript. As such, it cannot perform the same level of computational analysis or apply the same body of knowledge as a tool designed for guessing.

The second metric we apply is a guess-number calculator developed by Saranga Komanduri, which first appeared in Kelley *et al.* [12, 25]. We call this metric Weir+ because it builds on the guessing approach of Weir et al. [29]. Its advantages are that it is designed with the explicit goal of measuring the number of guesses required to crack a password, can be trained to target specific password policies, and represents the state of the art in measuring strength against a guessing attack. The disadvantages of Weir+ include that it is available only by contacting the author, written in multiple programming languages, and has not been made easy to configure. Further, its results may vary based on the size and quality of the training data. In order to create a large training set of passwords that comply with the Telepathwords policies, we used the 133,109 passwords in the Yahoo! Voices breach data set that received a score of 6 hard-to-guess characters or more from Telepathwords, which represents 29% of the 453,488 passwords revealed by that breach.

Our final metric is the score provided by the current version of Telepathwords itself—the number of hard-to-guess characters. We find this informative for comparing treatments *other* than those that employ Telepathwords. The scores for participants in telepath and telepath-v are provided exclusively for completeness, as participants who were able to generate a password that met the Telepathwords policy will score well by default (though we note that two participants received a 5 due to a change to predictions from the version deployed during the experiment and the version used to calculate scores).

Regardless of metric, the telepath and telepath-v passwords do substantially better than all other conditions, with the possible exception of 3class8-d. We present the scores for each metric in Table 8.

For the `zxcvbn` entropy measure, we show in Figure 16 that telepath and telepath-v passwords outperform those from all other conditions for the weakest password in each condition and the weakest 2.5%, 5%, and 10% of passwords. Thus, Telepathwords did the best job of preventing weak passwords. Only when we consider the median entropy do 3class12 and 3class8-d become competitive. The improvement with respect to 3class8 and basic8 is enormous.

Figure 17 illustrates the Weir+ measurements. Again the two Telepathwords conditions show enormous improvement over basic8 and 3class8. They show considerable improvement over 3class12 on minimum entropy, and on entropy of the weakest 2.5%, 5%, and 10%. The 3class8-d condition is roughly comparable to the two Telepathwords conditions, except when we consider minimum entropy, where it does considerably worse.

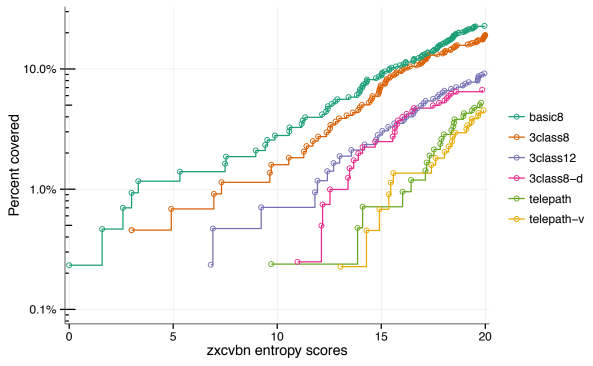


Figure 16: We sort the passwords in each condition by `zxcvbn`-entropy scores, from lowest to highest, and present the fraction of passwords with scores at or below a given value. Only passwords with entropy scores of 20 or less are shown in order to highlight the weakest passwords in each condition.

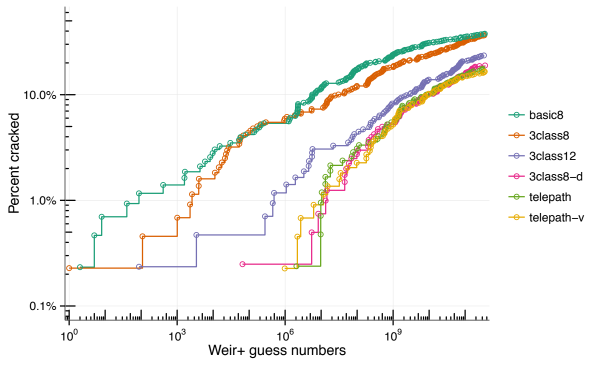


Figure 17: We sort the passwords in each condition by their Weir+ guess number, from lowest to highest, and present the fraction of passwords that with guess counts at or below a given number of guesses.

To substantiate further the impact of using Telepathwords and dictionary-based approaches, we present in Table 9 the weakest 2.5% of passwords according to each metric. For example, the weakest 2.5% under 3class8 contain such obvious and easily-guessed choices as `Password1` and `P@5sword`, which compare unfavorably with those in either of the Telepathwords conditions of 3class8-d.

		Mean characters per class				zxcvbn Entropy					Telepathwords score					Weir+ score			
	N	Upper	Lower	Digit	Symbol	Min	2.5%	5%	10%	Med	Min	2.5%	5%	10%	Med	Min	2.5%	5%	10%
3class12	425	1.7	8.0	3.2	0.9	6.8	14.8	17.0	20.4	33.4	2	3	3	4	7	6.4	22.4	27.8	31.5
3class8-d	402	1.5	6.6	2.6	0.8	11.0	15.6	17.8	22.1	32.7	1	3	3	3	6	16.0	26.6	29.4	33.6
3class8	438	1.6	6.6	2.5	0.6	3.0	11.7	14.1	16.1	29.1	1	2	2	3	6	0.0	14.4	17.4	24.9
basic8	429	1.0	7.9	2.4	0.4	0.0	9.5	12.6	15.4	27.9	1	2	2	3	6	1.0	13.0	17.7	22.4
telepath	420	1.0	7.0	2.5	0.6	9.7	17.9	19.7	22.4	32.0	5	6	6	6	7	21.0	26.1	29.3	33.0
telepath-v	441	1.1	7.0	2.7	0.5	13.0	18.4	20.4	22.4	32.8	6	6	6	6	7	19.9	27.4	29.9	33.2

Table 8: Security metrics of passwords created by participants. We show minimum and median `zxcvbn` and Telepathwords scores, along with percentiles selected to indicate the vulnerability of each condition to early guessing. We report Weir+ scores as the $\log_2$ of their guess numbers for comparison with entropy scores. We do not show median Weir+ scores as only basic8 reached 50% cracked in our analysis.

`zxcvbn`					
basic8	3class8	3class12	3class8-d	telepath	telepath-v
password	Password1	Thispassword1	1qaz2wsx!	thisisapassword	guessmypassw0rd
12345678	P@5sword	Password@123	123456789jI	2014welcome	Mary3476
P@55w0rd	EL1Z@B3TH	Qwerty12345@	Zaq12wsx	jim1965	altoids123
PASSWORD1234	Password8	Passwordneeds1	@bs0lute	$hrod3	almay123
passwordme	Mypassword1	!PaSsWoRd123	A11iance	1024scott	the1step!
sunshine	Samantha1	StephenASmith1	Beer4y0u	mothertrucker	snoopy1969!
Youknow123	Whatever1	1NewP@ssword	Hawk3y3s	burkeds	kylemonkey1986
brittany	Whoi1234	Chief$123456	G0dZ1Ll4	pi$$a123	Scr3wdr1v3r123
drowssap	My2password	MonKeY12345!	@SunSh1n3	12noraa	sion12
Washington1	Shelby1234	Asdfghjkl123	Cut13p13	c@reful951	lmi2014
					1987camaros
Weir+					
password	Password1	Asdfghjkl123	Pokemon91	1024scott	iamabeliever
12345678	Password8	Password@123	Redtruck1	jim1965	feefifofum
sunshine	Rainbow3	bulldog*1234	Nackson1	cesar5000	motuwethfr
brittany	Robert07	Jp1234567890	ZaqXsw12	mi1213	snorelax
qwertyuiop	Cougars1	Johnny#12345	H1r12345	mothertrucker	broseph
drowssap	Andrew24	Strawberry246	Monkeydude1	thisisapassword	cats59
trinity1	Marcus12	Guadalajara1	Plascencia1	imalittleteapot	peacaboo1
sugarbaby	Liverpool15	123Cheetos!!	Caedus12	awdxsz	almay123
deeznuts	Bahamut1	Abc123456789!	Godalmighty1	chieri	altoids123
monkey69	Abby1234	Qwerty12345@	Yaniku13	coffeecup123	jacran1
					sion12
Telepathwords					
frenchfry	Password1	MountainDew1	BearBear1		
qwertyuiop	EL1Z@B3TH	P00lsidebars	B4sk3r*v1ll3		
password	P@5sword	Elephants.19	Redtruck1		
p09op09o	Robert07	Password@123	A11iance		
P@55w0rd	Samantha1	cRAYON123456	Ilove!myself		
PASSWORD1234	Whatever1	MonKeY12345!	Zaq12wsx		
R0ckstar!	Qwaszx12	Abc123456789!	Cut13p13		
	Monkeys21	!PaSsWoRd123	Monkeydude1		
	Scoobydoo2	Asdfghjkl123	ZaqXsw12		
		Qwerty12345@	Galvestontx1		

Table 9: The weakest 2.5% of passwords as scored by each metric (weakest at top). For the Telepathwords metric, the weakest passwords in the telepathwords conditions are not shown because there are too many passwords at the minimum score threshold to present here.

5.4 Limitations

All artificial experiments have limitations and ours was no exception. We make note of two such limitations.

Our study used a role-playing scenario to encourage users to create passwords. Participants playing roles may choose weaker passwords than they would for an account they value. They might also choose stronger passwords than they would for an account they didn't value. Schechter *et al.* [22] have shown that participants in security studies behave differently when the laboratory environment frees them from risk, and Fahl *et al.* [7] have shown a specific effect for choice of passwords. While limiting the interpretations of absolute scores, so long as these effects impact conditions equally, the methodology still facilitates cross-condition comparisons—the primary focus of the experiment. In fact, if our goal is to study the ability of technology to help unmotivated users choose better passwords, having participants who are less motivated than they would be in real-world conditions may be beneficial.

We measured recall over a short period of two to five days in a context where participants entered their password a few minutes after choosing it. In contexts in which users do not re-enter their passwords immediately after creating them, or in which they do not return for more than five days, they may be more likely to forget them. In contexts where users use their passwords more frequently after creating them, they may be less likely to forget them within two to five days. Had we selected different return periods we might have been more likely to see differences in recall rates.

6 Related Work

While some security practitioners simply hope that passwords, and their associated weaknesses, can be wished away, Bonneau *et al.* [3] have argued that passwords are not going away anytime soon. Password-composition rules date back at least to 1979, when Morris and Thompson reported on the predictability of the passwords used by users on their Unix systems; they proposed that passwords longer than four characters, or purely alphabetic passwords longer than five characters, will be "very safe indeed" [19]. Bonneau analyzed nearly 70 million passwords in 2012, 33 years later, to measure the impact of a six-character minimum requirement compared with no requirement [2]. He found that it made almost no difference in security. In a study of the distribution of password policies, Florêncio and Herley found that usability imperatives appeared to play at least as large a role as security among the 75 websites examined [8].

Early studies of proactive password-quality verification mechanisms includes the work of Spafford [26], who suggests an efficient method for storing a dictionary for checking. Bishop *et al.*, in 1995, suggested checking passwords for dictionary entries, user information, and other common patterns at password creation [1]. They also provided some statistics on these patterns in passwords. Weir *et al.* also examined password-composition rules by looking at samples of passwords [29]. These works did not look at passwords created under varying rules, however.

Microsoft Windows has enforced password-composition rules at least as far back as 2000, with the default requiring at least 8 characters from three of four character classes: uppercase, lowercase, digits, and others [17, 18]. One problem with the Windows implementation is that when Windows rejects a user's proposed password, it does not provide a list of the rules being enforced or identify specifically which rules the password is violating.

Many websites offer password meters that provide feedback on the strength of passwords as users type them. Based on a survey of the top 100 websites in 2012, most password meters use simple password-composition rules such as length and number of non-lowercase characters to determine when a password is good enough to reach the next level on the meter [28]. Egelman *et al.* [6] examined whether the presence of a password meter made any appreciable difference in password strength. They found that the meter made a difference when users were changing their password for an existing important account; but the meter had little effect when users were registering a new password for a low-importance account. Ur *et al.* also studied the effect of password-strength meters on password-creation. They found that when users became frustrated and lost confidence in the meter, more weak passwords appeared [28]. Very recently, de Carné de Carnavalet and Mannan [4] examined several password meters in use at popular websites and found gross inconsistencies, with the same password registering very different strength across different meters. Collectively, these findings are in line with our concern that password policies and meters may harm credibility and lead users to put less effort into choosing a good password.

One exception to the reliance on composition rules in password meters is `zxcvbn`, an open-source meter developed and used by DropBox, which uses a small language corpus to calculate entropy estimates in real time [30]. Designed to run entirely in the users' browser, it is written in JavaScript and compresses down to 320KB. While `zxcvbn` provides a much-needed improvement in the credibility of its strength estimates when compared to approaches relying solely on composition rules, this credibility is unlikely to be observed by users. In fact, its perceived credibility may suffer if users, who have been told that adding characters increases password strength, see scores decrease when certain characters are added. For example, when typing `iatemylunch`, the strength estimate decreases from the second-best score (3) to the worst score (1) when the final character is added. Even if users find `zxcvbn`'s strength estimates credible, they are unlikely to understand the underlying entropy-estimation mechanism and thus be unsure how to improve their scores. The advice `zxcvbn` offers, such as using inside jokes and unusual use of uppercase, could potentially lead users to cluster around common strategies, yielding a set of new common passwords for attackers to guess.

Schechter *et al.* [21] offered another alternative to password-composition rules, suggesting a system that prevents users from choosing passwords popular among a large set of users. Another approach that seeks to limit dangerously common passwords was proposed by Malone and Maher [14]. These approaches, however, are most appropriate for systems with tens of millions of users, in which uniqueness is a strong indicator that a password is hard to guess. Relatively weak passwords may be unique among hundreds or thousands of user accounts.

The human-subjects experiment we perform in this work seeks to replicate the methodology used in prior password studies. Many of our choices in recruiting, question design, and the timing of the invitation to part two of the study reflect a desire to facilitate comparison with prior work. This includes the work of Komanduri *et al.* [13] and Kelley *et al.* [12], who used similar study designs to perform comparative analyses of password-composition rules. These prior studies found that increasing length requirements in passwords generally led to more usable passwords that were also less likely to be identified as weak by their guessing algorithm [13, 12]. Most recently, Shay *et al.* studied password-composition policies requiring longer passwords, finding the best performance came from mixing a 12-character minimum with a requirement of three character sets [25]. One key difference between our work and most prior studies is that all of our treatments provided feedback to users as they typed their passwords. With the exception of Ur *et al.*'s examination of meters providing optional guidance [28], all of these prior studies from Carnegie Mellon required participants to submit passwords *before* testing for, or providing feedback on, compliance with a policy.

A valuable use case for Telepathwords-based policies, which do not place any character-set requirements on passwords, is the affordance of creating all-lowercase passwords for easy entry on a touch screen. Jakobsson and Akavipat proposed a scheme for mobile devices that uses easily-typed passwords with auto-completion for easier password entry [11].

Fahl *et al.* [7] pointed out limitations in studies that use role-playing to generate passwords, as we do in this study. They find significant differences between passwords generated in these scenarios and real passwords. Komanduri *et al.* found that users created stronger passwords when asked to role-play, compared to when asked simply to create a password for a study [13]. Mazurek *et al.* used a methodology similar to ours and compared their results to genuine user passwords in a university [15]. They found that while the experimental passwords were slightly weaker than the genuine passwords, they were similar in many other respects.

7 Conclusion

Telepathwords provides users with significantly more insight into the quality of their passwords than all other approaches, and results in passwords stronger than approaches that do not use dictionaries. For example, the metrics suggest that to crack 1% of Telepathwords passwords, an attacker needs to make more than a factor of a thousand more guesses per password than for passwords created under the default password policy employed by Microsoft Windows Active Directory. While a higher number of users found password creation difficult or annoying using the tool, the security improvements did not come at any measurable impact to memorability.

References

[1] Bishop, M., and V Klein, D. Improving system security via proactive password checking. *Computers & Security 14*, 3 (1995), 233–249.

[2] Bonneau, J. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy* (May 2012).

[3] Bonneau, J., Herley, C., van Oorschot, P. C., and Stajano, F. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *2012 IEEE Symposium on Security and Privacy* (May 2012).

[4] de Carnavalet, X. d. C., and Mannan, M. From very weak to very strong: Analyzing password-strength meters. In *Network and Distributed System Security Symposium (NDSS14)* (2013).

[5] Designer, S. Openwall project free wordlist. `http://download.openwall.net/pub/wordlists/all.gz`.

[6] Egelman, S., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., and Herley, C. Does my password go up to eleven?: the impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), ACM, pp. 2379–2388.

[7] Fahl, S., Harbach, M., Acar, Y., and Smith, M. On the ecological validity of a password study. In *Proceedings of the Ninth Symposium on Usable Privacy and Security* (2013), ACM, p. 13.

[8] Florêncio, D., and Herley, C. Where Do Security Policies Come From? *Proc. SOUPS* (2010).

[9] How secure is my password. `https://howsecureismypassword.net/`.

[10] Hsu, B., and Ottaviano, G. Space-efficient data structures for top-k completion. In *22nd International World Wide Web Conferences* (May 13–17 2013).

[11] Jakobsson, M., and Akavipat, R. Rethinking passwords to adapt to constrained keyboards.

[12] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and López, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. *IEEE Symposium on Security and Privacy 0* (2012), 523–537.

[13] KOMANDURI, S., SHAY, R., KELLEY, P. G., MAZUREK, M. L., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND EGELMAN, S. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2011), CHI '11, ACM, pp. 2595–2604.

[14] MALONE, D., AND MAHER, K. Investigating the distribution of password choices. In *Proc. WWW* (2012).

[15] MAZUREK, M. L., KOMANDURI, S., VIDAS, T., BAUER, L., CHRISTIN, N., CRANOR, L. F., KELLEY, P. G., SHAY, R., AND UR, B. Measuring password guessability for an entire university. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 173–186.

[16] MICROSOFT CORPORATION. Check your passwordis it strong? `https://www.microsoft.com/en-gb/security/pc-security/password-checker.aspx`.

[17] MICROSOFT CORPORATION. TechNet Configuring Password Policies. Microsoft TechNet. `http://msdn.microsoft.com/en-us/library/cc236715.aspx`.

[18] MICROSOFT CORPORATION. Windows NT 4.0 Domain Controller Configuration Checklist. Microsoft TechNet. `http://technet.microsoft.com/en-us/library/cc722923.aspx`.

[19] MORRIS, R., AND THOMPSON, K. Password security: A case history. *Communications of the ACM 22*, 11 (1979), 594—597.

[20] The password meter. `http://www.passwordmeter.com/`.

[21] SCHECHTER, S., HERLEY, C., AND MITZENMACHER, M. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *The 5th USENIX Workshop on Hot Topics in Security (HotSec)* (Aug. 10 2010).

[22] SCHECHTER, S. E., DHAMIJA, R., OZMENT, A., AND FISCHER, I. The emperors new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *In Proceedings of the 2007 IEEE Symposium on Security and Privacy* (May 2007).

[23] SCHNEIER, B., AND KELSEY, J. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security 2* (1999), 159–176.

[24] SHAY, R., KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., UR, B., VIDAS, T., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Correct horse battery staple: exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (New York, NY, USA, 2012), SOUPS '12, ACM, pp. 7:1–7:20.

[25] SHAY, R., KOMANDURI, S., DURITY, A. L., HUH, P. S., MAZUREK, M. L., SEGRETI, S. M., UR, B., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Can long passwords be secure and usable? In *CHI* (2014).

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

[27] STARK, E., HAMBURG, M., AND BONEH, D. Symmetric Cryptography in Javascript. In *Annual Computer Security Applications Conference* (2009), pp. 373–381.

[28] UR, B., KELLEY, P. G., KOMANDURI, S., LEE, J., MAASS, M., MAZUREK, M. L., PASSARO, T., SHAY, R., VIDAS, T., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. How does your password measure up? the effect of strength meters on password creation. In *Proceedings of the 21st USENIX Security Symposium* (Aug. 8–10 2012).

[29] WEIR, M., AGGARWAL, S., COLLINS, M., AND STERN, H. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 162–175.

[30] WHEELER, D. zxcvbn: realistic password strength estimation. Dropbox Tech Blog. `https://tech.dropbox.com/2012/04/zxcvbn-realistic-password-strength-estimation/`, Apr. 2004.