Designing Password Policies for Strength and Usability

RICHARD SHAY, MIT Lincoln Laboratory†
SARANGA KOMANDURI, Carnegie Mellon University
ADAM L. DURITY, Google†
PHILLIP (SEYOUNG) HUH, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea
MICHELLE L. MAZUREK, University of Maryland, College Park
SEAN M. SEGRETI, BLASE UR, LUJO BAUER, NICOLAS CHRISTIN, and LORRIE FAITH CRANOR, Carnegie Mellon University

Password-composition policies are the result of service providers becoming increasingly concerned about the security of online accounts. These policies restrict the space of user-created passwords to preclude easily guessed passwords and thus make passwords more difficult for attackers to guess. However, many users struggle to create and recall their passwords under strict password-composition policies, for example, ones that require passwords to have at least eight characters with multiple character classes and a dictionary check. Recent research showed that a promising alternative was to focus policy requirements on password length instead of on complexity. In this work, we examine 15 password policies, many focusing on length requirements. In doing so, we contribute the first thorough examination of policies requiring longer passwords. We conducted two online studies with over 20,000 participants, and collected both usability and password-strength data. Our findings indicate that password strength and password usability are not necessarily inversely correlated: policies that lead to stronger passwords do not always reduce usability. We identify policies that are both more usable and more secure than commonly used policies that emphasize complexity rather than length requirements. We also provide practical recommendations for service providers who want their users to have strong yet usable passwords.

Categories and Subject Descriptors: D.4.6 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms: Human Factors, Security

Additional Key Words and Phrases: Passwords, password-composition policy, usable security, authentication

ACM Reference Format:
Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Designing password policies for strength and usability. ACM Trans. Inf. Syst. Secur. 18, 4, Article 13 (May 2016), 34 pages.
DOI: http://dx.doi.org/10.1145/2891411

---

†All work was performed while Richard Shay and Adam L. Durity were affiliated with Carnegie Mellon University. We previously reported on the results of the first experiment presented here in Shay et al. [2014].
This work is supported in part by NSF grants DGE-0903659 and CNS-1116776, and by a gift from Microsoft Research. This research was conducted with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.
Authors' addresses: R. Shay; email: rich@richshay.com; S. Komanduri; email: sarangak@cmu.edu; A. L. Durity; email: adurity@google.com; P. (Seyoung) Huh; email: one@etri.re.kr; M. L. Mazurek; email: mmazurek@cs.umd.edu; S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor; emails: {ssegreti, bur, lbauer, nicolasc, lorrie}@cmu.edu, lorrie@cs.cmu.edu.

ACM 1094-9224/2016/05-ART13 $15.00
DOI: http://dx.doi.org/10.1145/2891411

1. INTRODUCTION

To guide users toward passwords that are hard for attackers to guess, many service providers enforce password-composition policies. These policies put requirements on password creation, such as requiring that passwords contain a minimum number of characters, use particular character classes, or not be contained in a list of common passwords. However, when these requirements become too onerous, users struggle to create and remember their passwords [Komanduri et al. 2011].

For example, Carnegie Mellon University uses a "comprehensive" password policy that requires eight characters, four character classes, and includes a dictionary check.[1] Our prior research found that users struggled to create and recall passwords under this comprehensive policy [Komanduri et al. 2011]. We also found that password creation and recall were easier, and passwords were overall significantly more difficult to guess, under an alternative policy that only requires that passwords be at least 16 characters long. However, a small, but nonnegligible, number of participants created very easily guessed passwords under this alternative "longer password" policy, such as *passwordpassword* [Kelley et al. 2012; Komanduri et al. 2011]. As a result, this simple policy based on longer passwords, despite its usability and some security advantages, may not be quite suitable for real-world deployment.

In this article, we describe our search for password-composition policies that we can recommend to service providers. Specifically, we ran two large user studies and examined 15 password-composition policies, testing several permutations of length and character-based requirements. We compared the strength of passwords created under each policy, as well as user sentiment, timing, and attempts required for password creation and recall. The first study examined 13,751 passwords created by participants under one of eight policies. To confirm the previous results that suggested that length requirements alone are not sufficient for usable and secure passwords, in addition to the comprehensive 8-character policy earlier described, we tested policies requiring 12, 16, and 20 characters without further requirements. We also tested adding a three-character-class requirement to the length-12 and length-16 requirements. These policies attempted to strike a balance between comprehensiveness of the policy and length requirements. This study showed that certain combinations of character requirements and longer-length requirements led to fewer easily guessed passwords than a policy purely based on length, while still being more usable and more secure than a comprehensive policy with shorter length requirements.

Several observations from data collected during the first study led to hypotheses that we tested in the second study. We noticed that certain substrings appeared to be markers of weak passwords. For example, passwords containing the substring *1234* were twice as likely to be guessed as those that did not. Hence, we studied passwords collected under a *blacklist* requirement, which prohibited passwords from containing any substring from a specified list. Further, we observed that passwords that were required to have multiple character classes rarely began and ended with lowercase letters. Instead, participants tended to put special characters at the start or end of their passwords. Those passwords that did begin and end with lowercase letters were quite strong. This observation led us to test a *pattern* requirement—the requirement that passwords begin and end with lowercase letters.

The second study presented in this article examines eight password policies, including policies based on the blacklist and pattern requirements. We collected data from 8,740 participants and found further evidence that a password policy requiring 12

[1]http://www.cmu.edu/iso/governance/guidelines/password-management.html.

characters and two or three character classes is more usable and more secure than a comprehensive policy.

We also found evidence that while users often meet longer-length password requirements in predictable ways, policies can break users of undesirable password-creation habits. The blacklist requirement succeeded in leading users to create significantly stronger passwords. While the blacklist requirement also made password creation more difficult, it did not affect password recall. Overall, the blacklist requirement offers a very favorable tradeoff between security and usability for a number of real-world use cases. The pattern requirement induced a more even character-class distribution within passwords. This led to a significant increase in password strength, even over passwords with the blacklist requirement. While the pattern requirement did make passwords more difficult to create and recall, we believe that the substantial improvement in strength means the pattern requirement may be well suited to highly sensitive environments.

In sum, this work makes contributions to both researchers and system administrators, who previously had to rely on educated guesses rather than scientific data when selecting password-composition policies [Burr et al. 2011].

We conduct large-scale human-subjects testing on both deployed and new password-composition policies. The comprehensive data we collect about the security and usability of these policies leads us to recommend three policies in place of the more traditionally deployed comprehensive policy based on multiple requirements. All three of the policies we recommend strike a balance between length and character-class complexity in their requirements. We also find that requiring that passwords begin and end with a lowercase letter leads users to place digits and symbols in less predictable places in their passwords. Furthermore, we find that blacklisting common password substrings leads to passwords that are less predictable by attackers without being less memorable to users, and we propose an algorithm for creating such a password blacklist from a password corpus.

We continue with a discussion of previous research in Section 2. We describe the methodology behind our studies and discuss its limitations in Section 3. We present findings from our first study in Section 4. In Section 5, we discuss how the findings from our first study led to the conditions in our second study. We then present the findings of the second study in Section 6. In Section 7, we compare two pairs of salient conditions from across the two studies. Finally, in Section 8, we discuss the implications of our studies, and practical advice for service providers that they lead to.

2. BACKGROUND AND RELATED WORK

We next present background and related work to provide context for the reader. Section 2.1 discusses the offline attack model, which is the threat model on which we focus. Section 2.2 discusses prior research on password-composition policies in order to show how our work fits into the larger passwords-research landscape. Because our studies collected data online, Section 2.5 discusses prior online password-policy research. Finally, Section 2.6 discusses how previous passwords research has motivated our own focus on longer passwords.

2.1. Threat Model: Offline Attack

The work in this article is intended to help service providers keep users' online accounts safe from hijacking through password-composition requirements that are usable, and that lead to secure passwords. Account hijacking can lead to financial harm, as well as emotional distress and embarrassment [Shay et al. 2014]. Despite the shortcomings of text passwords, none of the immediate candidates to replace them appear to offer all of their advantages [Bonneau et al. 2012]. Researchers have argued that no proposed replacement is likely to replace them as the entrenched standard any time soon [Herley and Van Oorschot 2012].

In an *online* attack against a password-protected account, the attacker tries guessing the victim's password on a live system. The service provider can lock out the attacker after a fixed number of failed log-in attempts over a short period of time. However, an attacker can avoid detection by spreading out the guesses over time [Florêncio et al. 2014].

This article focuses on the *offline* attack model, in which an attacker steals a hashed password file from a service provider. Theft of hashed password sets is occurring frequently enough to make the news many times per year, affecting millions of users [Perlroth 2013; Engberg 2013; Lord 2013; Kumparak 2013; Baraniuk 2015; Zheng 2015].

We assume that the passwords in the password file are both salted and hashed [Florêncio and Herley 2010]. A salt is a string, unique to each user, that is added to each password before being hashed. A hash is a one-way function which, when given a (salted) password as input, deterministically produces an output string that cannot be used to determine the input string. A salt prevents two users who share the same password from having the same hash.

Attackers in an offline attack typically generate a password guess candidate and apply the same salt and hashing function as the targeted passwords in the stolen file. Salts are generally stored in clear text near the corresponding password. If the result of hashing the guess candidate matches a victim's hash, then the attacker knows the guess candidate was correct. If not, the attacker can continue making and hashing guesses, bounded by time and hardware. An optimal attacker will guess the most likely, or expected, passwords first. Therefore, an attacker's success depends on the relative unpredictability or *strength* of the victim's password. Password-guessing algorithms can enable attackers to leverage existing password corpora to generate password guesses in probability order, as we will describe in greater detail in Section 3.2.

An attacker conducting an offline attack hashes each password guess. While many common hashing algorithms are designed to execute quickly, there are also hashing algorithms that are deliberately designed to be slow in order to hobble attackers conducting an offline attack. The bcrypt hashing scheme, for example, can be configured with a cost factor that exponentially increases its execution time by requiring a sequential series of computations [Provos and Mazieres 1999]. Other password-hashing schemes, including scrypt [Percival 2009] and Argon2 [Biryukov et al. 2015], rely on sequential memory-hard functions. Because a service provider can use a slower hash function to increase the time each guess requires for an attacker, we present our security results in terms of the number of guesses, as opposed to wall-clock time.

An important question in the password ecosystem is the degree to which one should expect users to shoulder the burden of password security by creating hard-to-guess passwords, as opposed to system administrators following best practices. The ability of a password to withstand a large number of adversarial guesses matters far less for a system that secures passwords using one of the password-hashing schemes described above. However, recent password breaches show that websites still often use computationally inexpensive hash functions like MD5, making large-scale offline attacks feasible. For instance, the 2015 breach of educational technology company VTech revealed that their passwords were stored using MD5 [Zheng 2015]. While initial reports about the 2015 breach of Ashley Madison, a dating site for married individuals, revealed that passwords were stored using bcrypt, a legacy database was later discovered; the legacy database contained many of the same passwords hashed using MD5 [Goodin 2015]. These events suggest that a security-conscious user should not rely exclusively on websites following best practices for storing passwords. Consistent with the idea of defense in depth, a rational user should also make a password that is hard for attackers to guess.

Other technical mechanisms aim to protect the password databases themselves. In order to make password files themselves more difficult to steal, researchers have proposed distributing authentication across multiple servers [Camenisch et al. 2015; Ford and Kaliski Jr 2000]. However, despite such a scheme being proposed in 2000, it has not seen widespread adoption, likely because it requires additional hardware and server coordination. Researchers have also proposed schemes that make offline attacks easy to detect using decoy passwords [Juels and Rivest 2013]. While such a scheme helps a service provider quickly detect an attack and thereby minimize potential damage, it does not directly help users who have reused the same password across sites. Therefore, while improving server security in general is a crucial long-term goal, helping users make stronger passwords is an immediate step that can help to protect users.

2.2. Password-Composition Policies

Many service providers have password-composition policies intended to prevent users from creating easily guessed passwords. These policies constrain the space of allowed passwords, attempting to prevent users from making passwords that an attacker would guess easily. An ideal password-composition policy helps users make strong passwords, without making it onerous to create and recall passwords.

Without a password-composition policy to force them to make stronger passwords, research has shown that many users will opt to create simple, easily guessed passwords. In a 1995 study, researchers requested that system administrators send them hashed passwords. The researchers were able to crack about 40% of the approximately 14,000 passwords using a dictionary attack. This led to a call for "proactive password checking" to make sure passwords comply with a set of password-composition requirements [Bishop and Klein 1995]. The National Institute of Standards and Technology (NIST) has also advocated instituting password-composition requirements to force users to make stronger passwords, noting that users often make passwords as simple as they are allowed [Burr et al. 2006]. More recently, we found that study participants required to create passwords with no requirement other than having eight characters created passwords that were significantly weaker than participants with more strict password-composition requirements [Kelley et al. 2012].

Password-composition policies need to consider both the security of the passwords created under them, and the usability of those passwords. In a study of over 25,000 members of the Carnegie Mellon University community, users who were annoyed by the university's password-composition policy made weaker passwords [Mazurek et al. 2013]. A 32-participant diary study, published in 2010, observed that password-composition policies were often burdensome to users and could lead to decreased productivity [Inglesant and Sasse 2010]. Multiple interview studies have suggested that users struggle to create strong passwords [Stobert and Biddle 2014; Ur et al. 2015a]. These studies indicate that the password-composition policies intended to protect users can also impose a burden on them. In addition to negative sentiment, user burden can lead to help-desk calls and password reuse [Herley 2009].

Password-composition policies are intended to make it difficult for attackers to guess passwords by making passwords less predictable. However, their effectiveness is often limited because users tend to fulfill their requirements in predictable ways [Burr et al. 2011]. For example, one study found that users often select symbols from only a small fraction of the symbols on a keyboard [Shay et al. 2010]. Another study found that when users employ digits, symbols, and uppercase letters in their passwords, they often do so predictably [Weir et al. 2010]. Users tend to make passwords that are semantically meaningful [Veras et al. 2014] and follow grammatical rules [Rao et al. 2013; Ur et al. 2015a]. Furthermore, passwords often contain predictable phrases [Bonneau and Shutova 2012; Ur et al. 2013], dates [Veras et al. 2012], and common types of words [Veras et al. 2014].

2.3. Understanding How Users Manage Passwords

Our work focuses on contrasting different password-composition policies. Prior work has focused on understanding how users treat their passwords. That work provides context for our own, and may help to explain our participants' behavior.

Many studies have found that password reuse is common. Based on analyses of leaked datasets, researchers have concluded that around half of users reuse the same password across different websites [Das et al. 2014]. In an interview study of 27 participants' strategies for password usage and management, researchers found that participants used an average of 11 different accounts in a week yet had an average of only five unique passwords. Although most participants used software like Apple Keychain to help manage passwords, none used a dedicated password manager [Stobert and Biddle 2014]. Security experts also frequently reuse passwords, yet often take into account the security risks of doing so [Stobert and Biddle 2015]. Researchers have used theoretical modeling to suggest that some degree of password reuse is a rational behavior by users for low-value accounts. They argue that weak passwords and password reuse are behaviors that are unlikely to change [Florêncio et al. 2014].

Password reuse and other coping strategies, such as writing passwords down, are necessary because users have dozens of accounts. In a large-scale, in-situ study of nearly 500,000 participants who installed a Windows toolbar, researchers found that users have an average of 25 accounts and 6.5 different passwords. Most of these passwords were only lowercase letters, except for accounts that required multiple character classes [Florêncio and Herley 2007]. A 2013 study of 2,000 United Kingdom adults found participants had an average of 19 accounts [Experian 2014].

Users' mental models are a crucial driver in the management of passwords. A number of researchers have found that some, but not all, users believe different accounts have different value and importance, impacting how they create and manage passwords for those accounts [Ur et al. 2015a; Stobert and Biddle 2014, 2015]. As evidenced by a series of 33 interviews investigating how users understand online threats, users often have a poor understanding of threat models. This poor understanding of why or how attackers would access their data might lead to users making suboptimal security-related decisions [Wash 2010]. Security experts' mental models of passwords have been found to differ from nonexperts' mental models; experts felt losing a password was analogous to losing a credit card number, whereas nonexperts compared losing a password to losing a physical key [Asgharpour et al. 2007].

2.4. Measuring Password Strength

Earlier research on passwords has used estimated password entropy to measure password strength [Florêncio and Herley 2007; Shay et al. 2010]. The uneven distribution of the predictability of user-created passwords, however, makes the entropy of an entire password set a less useful metric. More recently, researchers have argued against using entropy, preferring to use metrics likes "marginal guesswork" [Pliam 2000]. This metric examines the number of guesses required to guess a given password, for a given way of generating guesses. Furthermore, the entropy of a password set does not sufficiently convey how difficult it is for an attacker to guess passwords from that set. Researchers have presented statistical techniques for modeling how much of a password set withstands attacks of different strengths, demonstrating these techniques on a very large corpus of passwords [Bonneau 2012]. While excellent for large corpora of passwords, these statistical techniques do not apply to small datasets and do not reflect real-world attackers. As a result, we present password strength as a function of how many guesses a password requires to be guessed under given password-cracking approaches using particular training data [Komanduri 2016]. We do so using our group's Password Guessability Service (PGS) [Carnegie Mellon University 2015], which we have shown in separate work to be a conservative proxy for an expert, human attacker [Ur et al. 2015b]. We detail how we calculated guessability in Section 3.2.1.

A number of researchers have studied how to compare password sets efficiently. Ma et al. [2014] proposed comparing sets using probabilistic password-cracking approaches (e.g., a probabilistic context-free grammar or a Markov model) by focusing on probabilities, rather than the number of guesses a particular password-cracking approach takes to arrive at a given password. Recently, researchers demonstrated an efficient way to estimate the number of guesses a given probabilistic algorithm would need to guess a password. Their technique uses Monte Carlo simulation and can be adjusted to trade off speed for accuracy [Dell'Amico and Filippone 2015].

2.5. Prior Mechanical Turk Password Studies

Our research group has used Amazon's Mechanical Turk crowdsourcing service (MTurk) to study password policies in previous studies. In prior MTurk-based password studies, we found evidence that asking participants to role-play results in significantly stronger passwords, suggesting that participants are more invested in the passwords they are creating. Compared to participants who were asked to create passwords as part of a study, participants who were asked to imagine that they were creating a password for their real email account made significantly stronger passwords [Komanduri et al. 2011]. This result suggests that asking participants to role-play makes them treat study passwords as more real, even though study passwords objectively have no actual value to a participant.

In our prior MTurk studies, we found that password-composition policies can affect both usability and security. One policy we studied requires eight characters, four character classes, and a dictionary check. Compared to a more lax policy that only required eight characters, passwords created under this comprehensive policy were much less likely to be guessed. However, they were also more difficult to create and recall [Kelley et al. 2012]. We studied system-assigned passwords and passphrases and found that users struggled with both. Three- and four-word system-assigned passphrases did not offer usability advantages over system-assigned passwords of equal entropy, and were often less usable [Shay et al. 2012]. We also studied password-strength meters and found they can lead to significantly stronger passwords. While the visual depiction of password strength did not appear to make a difference, having a more stringent meter led to stronger passwords [Ur et al. 2012]. In addition, we conducted a study focused on giving users real-time password-creation feedback. We found that giving participants feedback on whether and how their passwords met requirements reduced password-creation errors without reducing password strength [Shay et al. 2015].

A research group unaffiliated with our own recently used a similar methodology to study passwords. They used MTurk to collect data from over 5,000 participants, using a two-part methodology that follows our general data-collection outline. Participants were either assigned a password, created their own password, or were assigned a password and asked to modify the password. They found that allowing users to modify system-assigned passwords made those passwords more memorable, yet less secure. They also found evidence that modified system-assigned passwords were more secure than user-created passwords, and in some cases were as memorable [Huh et al. 2015].

Any online, role-playing study raises questions of ecological validity. To understand better the ecological validity of MTurk studies about passwords, we compared the single-sign-on passwords of approximately 25,000 Carnegie Mellon University faculty, staff, and students to sets of passwords created by MTurk workers [Mazurek et al. 2013]. In particular, we asked MTurk workers to create passwords under CMU's password-composition policy on a sequence of web pages that mimicked the actual CMU pages. Unlike the CMU affiliates' passwords, the MTurk workers' passwords had no actual value. We found that the MTurk workers did create passwords that were slightly weaker than the genuine ones. Surprisingly, however, they were relatively close in strength. In addition, the two sets of passwords shared many characteristics, including structure and composition. This finding provided evidence that MTurk is a valid platform for conducting research into the effects of password-composition policies. Researchers from another university similarly studied the ecological validity of research studies about passwords. Participants from their university created passwords for either an online study or a laboratory study, and the researchers compared these passwords to those users' actual passwords for their university account [Fahl et al. 2013]. While they found laboratory studies to result in study passwords that are more representative of participants' actual passwords, only one third of passwords from the online study were not representative of participants' actual university passwords.

2.6. Motivation for Studying Longer Passwords

One promising alternative policy that we identified in our prior work was to require only that passwords contain at least 16 characters. We found that password creation and recall were easier and less error-prone under this alternative policy than under the more complex comprehensive policy described earlier [Komanduri et al. 2011; Kelley et al. 2012]; and that, overall, passwords were also less likely to be guessed by an attacker making a large number of guesses (10^{12}). However, under this alternative 16-character policy, some participants created simple, easily guessed passwords. As a result, an attacker with the ability to make only 10^8 guesses would be able to crack more passwords under the 16-character policy than under the comprehensive 8-character policy.

Because of these easily guessed passwords, we did not feel comfortable recommending the otherwise-promising longer password policy. In this paper, we advance the state of the art in password-policy research by finding policies that can be recommended for use in practice. We find policies that are more usable than the comprehensive policy we have seen deployed in practice, are as secure against a resource-constrained attacker, and more secure against an attacker capable of a larger number of guesses.

3. METHODOLOGY

We conducted a pair of two-part online studies to examine how participants create and use passwords. In this section, we describe our data-collection protocol, how we analyzed that data, and the limitations to our approach. Our two studies had different conditions but otherwise used the same methodology. We used MTurk to recruit and pay participants. We assigned participants at random to conditions because we wanted to draw causal conclusions to make specific policy recommendations.

In the first part of each study, *Part 1*, we asked participants to create a password under a given policy. Participants then filled out a brief survey and recalled their password. Two days later, we invited participants to return for the second part of the study, *Part 2*. We asked them to recall their password again, and to take a second survey. We paid participants 55 cents for completing Part 1 and 70 cents for completing Part 2. Previous passwords research has used a similar data-collection methodology [Komanduri et al. 2011; Kelley et al. 2012; Shay et al. 2012; Ur et al. 2012].

3.1. Study Overview

Participants needed to be at least 18 years old and located in the United States, as determined by MTurk. In Part 1, we asked participants to imagine that their email provider had been compromised and required they create a new password, under a policy determined by their condition. We informed participants that we would ask them to return and recall their password again in a few days. We asked participants to take whatever steps they normally would to remember and protect their passwords. Prior work showed that asking participants to imagine creating email passwords leads to stronger passwords than just asking them to create study passwords [Kelley et al. 2012].

After creating and confirming a password, participants completed a 5-minute survey about their experience. We then asked participants to recall their password, which we call *Part 1 Recall*. If participants did not enter the password successfully in five attempts, we showed it to them.

Two days later, we invited participants to return for Part 2 of the study. We did this using an MTurk feature that allowed us to email participants without knowing their email addresses. Once they returned, we asked participants to recall their passwords, which we call *Part 2 Recall*. Participants who entered five incorrect passwords saw their password on the screen. Participants could also follow a “Forgot Password” link to be emailed a link to their password. After Part 2 Recall, we administered a 5-minute survey about whether and how participants stored their passwords.

3.2. Measuring Password Guessability

In order to evaluate password guessability across conditions, we simulated an offline attacker with a stolen salted-and-hashed password file. The attacker attempts to guess a user’s password using a particular password-cracking algorithm configured with particular training data. For each guess, the attacker salts and hashes that guess. If the hash of the guess matches the entry in the password database for a particular user, the attacker has discovered that user’s password.

We calculated how many guesses were required to guess a given password in two ways, and we conservatively took the smaller guess number from these two methods. The first method was a probabilistic context-free grammar (PCFG) password-guessing algorithm [Kelley et al. 2012; Weir et al. 2009]. The second method was the CMU Password Guessability Service (termed *PGS*) [Carnegie Mellon University 2015]. We explain both below.

First, we ran an attack targeted to each particular set of passwords created by our participants under a given password-composition policy. We term this method a *targeted PCFG*, a type of PCFG password-guessing algorithm. The PCFG approach uses training data to generate guesses in order of likelihood, up to a probability minimum cutoff. For each study condition, we randomly divided its passwords into two folds, and we used each fold to help train the PCFG algorithm for cracking the other fold [Shay et al. 2014; Kelley et al. 2012].

For our targeted PCFG approach, we improved the PCFG algorithm to help it crack the longer passwords that are the focus of this work. The original algorithm could only guess a password containing a long string of letters if that exact string was found in the training data. For example, unless training data contained the string *desktopbackup*, any password containing that string would not be guessed. To address this, we augmented the grammar of the PCFG algorithm to include word breaks. Thus, a password containing *desktopbackup* could be guessed if the training data contained *desktop* and *backup* as separate strings. We tokenized passwords using a word-level, frequency-based n-gram model, where an n-gram model is used to determine separation points in unbroken text [Komanduri 2016]. Using this model, we break up long alphabetic strings into individual words.

We also adjusted the PCFG algorithm to include the frequencies of letter strings so that passwords with more common letter strings be found first. The original PCFG implementation assigned letter strings uniform frequencies for performance reasons [Weir 2010]. Our refinement improves the effectiveness of the guessing algorithm by favoring high-probability strings but also increases computational and memory requirements due to the large increase in number of training strings. We mitigate this by quantizing probabilities, trading accuracy for speed [Komanduri 2016]. Groups of strings with the same frequency are packed together into a data structure and treated as a single unit, greatly increasing speed. We use the Lloyd-Max algorithm, which minimizes the mean squared error of the quantization, so accuracy is not reduced too much [Komanduri 2016].

We configured the targeted PCFG approach to generate only guesses that conform to the length, character-class requirements, and pattern requirements of the password-composition policy. Because we perform our guessability analyses using code that implements the PCFG as a lookup table [Kelley et al. 2012], rather than a list of enumerated guesses, we were unable to configure the PCFG approach to avoid guesses that match a dictionary or blacklist. Thus, conditions with dictionary or blacklist checks might appear slightly more resistant to guessing than they would be in practice.

Second, we ran each set of passwords through the PGS, a public service for researchers that our group released based on evaluating numerous configurations of different approaches to password cracking [Ur et al. 2015b]. In particular, our PGS guess numbers conservatively represent the smallest guess number across three approaches to password cracking: an order-5 Markov model [Ma et al. 2014]; the password-cracking tool oclHashcat [Steube 2015]; and a PCFG [Kelley et al. 2012; Weir et al. 2009] that differs from the one described above by not using cross-validation, yet using a more recent PCFG implementation that considers unseen terminals [Komanduri 2016] and also guesses passwords from the training data before abstracting them into a grammar [Ur et al. 2015b]. Guess numbers generated by a PCFG and Markov model only generate guesses that comply with the length and character-class requirements of the password-composition policy. In contrast, Hashcat guess numbers are not filtered by password-composition policy because the oclHashcat software tool does not offer such filtering in the modes in which it is used for PGS [Ur et al. 2015b].

3.2.1. Modeling the Attacker. We trained our targeted PCFG using publicly available data and more targeted data. The publicly available data included the Google web corpus [Brantz and Franz 2006] and a free password-cracking dictionary from the Openwall Project.[2] We trained on two publicly available leaked password sets: the RockYou set of over 30 million passwords [Vance 2010] and the MySpace set of about 45,000 passwords [Schneier 2006]. We also trained on passwords created in one of our prior studies [Kelley et al. 2012]. We collected them on MTurk using a similar methodology. An attacker could conduct similar data collection to make a similar training set.

The PGS attacks were also trained using leaked passwords and dictionaries. The passwords were taken from breaches of MySpace [Schneier 2006], RockYou [Vance 2010], and Yahoo [Goodin 2012]. In addition, PGS uses the natural-language dictionaries found most effective in prior work [Weir et al. 2010; Kelley et al. 2012]: all single words in the Google Web corpus [Brantz and Franz 2006], the UNIX dictionary [Beeman 2004], and a 250,000-word inflection dictionary [SCOWL 2015]. For cracking approaches that take only a wordlist, without frequency information, PGS

[2]http://www.openwall.com/wordlists/.

uses a wordlist ordered by descending frequency and with duplicates removed. To generate additional guesses for the Hashcat attacks, we augmented this wordlist with the wordlists provided by InsidePro [InsidePro 2005].

For our resource-intensive targeted PCFG attacks, we were able to simulate 10^{12} guesses in each condition. For PGS, the guessing cutoff differs by password-cracking approach, depending on the resource requirements per approach, but exceeds 10^{14} guesses for the most computationally efficient approaches. As a result, the PGS guess numbers reflect 10^{14} guesses for each condition.

One way we compared password strength between conditions was looking at how many passwords in each condition were guessed after a certain number of guesses. We use two guess-number cutoffs: 10^6 and 10^{14} guesses. Florêncio et al. [2014] have suggested these as guess-number thresholds for offline and online attacks, respectively. Previous research suggests that some policies that perform well against an attacker making a larger number of guesses might perform comparatively poorly against an attacker making a smaller number of guesses and vice-versa [Kelley et al. 2012]. Our offline guess-number cutoff is optimistic and assumes that the service provider is following good practices, including using a slow hash function, monitoring for data breaches, and forcing password resets if a breach is detected. If the service provider is not following these practices, then an attacker might make a larger number of guesses.

To provide a rough idea of how these guess numbers translate to time, we performed benchmarking tests in our lab. We used a standard laptop computer[3] to perform hashes with the deliberately slow *bcrypt* password-hashing function [Provos and Mazieres 1999]. We hashed 1,000 passwords of eight random lowercase letters each. This took 73.3 seconds. In order to reach 10^{12} guesses, it would take the attacker over 2,000 years. We used the default setting for bcrypt; increasing the cost factor parameter would have made this take longer. In comparison, using MD5 to hash 10^{12} passwords on the same computer would take 55.9 days.

3.3. Limitations

Next, we discuss limitations of our protocol and analysis. We tested recall only after a few minutes and a few days. In practice, users can utilize passwords in a variety of patterns. Real-world users might not use their passwords for a long period of time or may be using them multiple times per day. Thus, our study only covers a subset of real-world use-cases. We examined only a limited number of password-composition policies. Furthermore, we used only one particular set of training data to calculate password guessability. While prior work has shown this particular set of training data to support guessability analyses that are a conservative proxy for an experienced attacker [Ur et al. 2015b], a new set of attack techniques or novel set of training data could prove to be substantially more effective against some or all of the password-composition policies we test.

Unlike real-world, high-value passwords, study participants would not suffer negative consequences if they chose a weak password. Although we requested that participants treat their study password as if it were the password for their main email account, they were not otherwise incentivized to remember their study passwords or to treat these passwords as genuine.

As described in Section 2.5, in prior work we found that MTurk workers created similar passwords to genuine Carnegie Mellon University passwords [Mazurek et al. 2013]. Further, we found evidence that asking users to role-play during password creation appears to increase buy-in, as participants make stronger passwords [Komanduri et al.

[3]2012 MacBook Pro, running OSX 10.10.1 (Yosemite), Retina display, 8GB 1,600MHz Memory, with a four-core single-processor 2.3GHz Intel Core i7, running ruby 1.9.3.

2011]. Another study recruited participants from a German university and compared study passwords with genuine university passwords. Those researchers concluded that 46% of the passwords from the online study were fully representative of those users' actual passwords, and another 23% were partially representative [Fahl et al. 2013]. These studies provide evidence for the ecological validity of online passwords research.

One realistic advantage of a novel password policy is that an attacker has less training data. However, this benefit may be temporary. If more service providers adopted the novel policy, more passwords created under that policy would appear in leaked password sets, giving attackers more training data. In our research, we were practically limited in the training sets and the password-cracking algorithm we used. It is likely that future attackers will have access to additional leaked password sets and more advanced password-cracking algorithms.

Our study had a high number of participants who began but did not complete both parts. We cannot be sure why participants opted not to complete the study. Some participants may have missed our emails inviting them to return, and others might have been confused, bored, or frustrated by the study. We discuss study dropout and completion rates in Sections 4.4.1 and 6.3.1. We observe cases where there are statistically significant differences across conditions for dropout rates, which we interpret as evidence of usability differences across conditions.

Finally, there were study factors beyond our control. For instance, we did not control the device or keyboard used to input passwords. Prior work has indicated that input devices do make a difference in password strength [von Zezschwitz et al. 2014; Yang et al. 2014; Melicher et al. 2016]. Password creation on mobile devices takes more time and results in more errors and greater user frustration [Melicher et al. 2016]. Mobile keyboards may be substantially different from standard desktop keyboards. There is also a considerable ranges of styles and usability among mobile-device keyboards [Schaub et al. 2012]. As a result, our results may not apply fully to passwords created on mobile devices.

3.4. Statistical Testing

Our statistical tests used significance level $\alpha = .05$. For omnibus comparisons on quantitative data, we used Kruskal-Wallis (KW). For omnibus comparisons on categorical data, we used Chi-Squared (χ^2). If the omnibus test was significant, we performed pairwise tests with Holm-Bonferroni correction (HC). We used Mann-Whitney U (MW) for pairwise quantitative comparisons and Fisher's Exact Test (FET) for pairwise categorical comparisons. These tests do not assume a normal distribution among our data. Whenever this text calls a comparison significant, it indicates statistical significance. All time-based statistical comparisons use medians instead of means.

4. STUDY 1

This section presents the conditions and findings of our first study. We found two password policies that were more usable and more secure than the shorter comprehensive policy. These policies both combined longer-length and character-class requirements. We also observed patterns among easily guessed passwords that motivated the second study presented in this paper. Table I summarizes the findings for this first study.

4.1. Conditions

comp8—We asked participants to include "at least 8 characters," including a "lowercase English letter," "uppercase English letter," "digit," and "symbol (something that is not a digit or an English letter)." We also told participants, "Taken together, the letters must not form a word in our dictionary." We used the free Openwall password-cracking

Table I. A Summary of Findings for Study 1

Condition	Part 1 completion (%)	Password storage (%)	Mean creation attempts	Agree creation difficult (%)	Part 2 recall attempts	Part 2 entry time (s)	Agree remembering difficult (%)	Cracked@10^6 (%)	Cracked@10^{14} (%)
comp8	**83.0**	**56.9**	**2.4**	**32.8**	**1.7**	**13.2**	**39.3**	**2.2**	**50.1**
basic12	94.5	45.4	1.5	15.2	1.6	11.6	27.4	9.1	52.0
basic16	93.9	49.9	1.8	28.5	1.6	13.7	30.1	7.9	29.7
basic20	93.9	50.0	1.9	35.2	1.6	15.3	32.9	5.6	16.4
2word12	92.0	51.4	1.9	21.9	1.6	13.1	31.0	3.4	46.6
2word16	92.1	51.3	2.1	34.7	1.7	14.6	36.8	1.1	22.9
3class12	92.0	54.9	1.5	26.0	1.7	14.8	35.3	3.2	36.8
3class16	90.5	60.2	1.9	40.3	1.7	16.2	42.9	1.2	13.8

Each condition is compared to comp8. Light blue indicates being statistically significantly better than comp8, and dark red indicates being worse. No shading indicates no statistically significant difference.

dictionary[4] for the dictionary check. We removed digits and symbols from the password and checked it against this dictionary, case-insensitive. This condition is similar to a traditional strong password policy. We wanted to find other policies that would perform better in either security or usability, without being worse in any metric.

basic12, basic16, basic20—We asked participants to include at least 12, 16, or 20 characters. b16 was previously found to be more usable than comp8, but also to allow more easily guessed passwords [Kelley et al. 2012]. We wanted to replicate that result with basic16 and also to examine both longer and shorter length-only requirements.

3class12, 3class16—We asked participants to include at least 12 or 16 characters, and at least three of the four character classes. These conditions combined longer length requirements with a subset of the character-class requirements of comp8. We hypothesized that these conditions might combine the greater usability of longer passwords while preventing easily guessed passwords that would be created under length-only requirements.

2word12, 2word16—We asked participants to include at least 12 or 16 characters and to have "at least two words (letter sequences separated by a nonletter sequence)." We hoped to encourage passphrases, which can be more memorable than traditional passwords [Keith et al. 2009]. These conditions combined longer-length requirements with fairly minimal character-class requirements, and we hoped they would be more secure and more usable than comp8.

4.2. Participants

We recruited participants between April and June 2013. Table II shows the number of participants per condition. Of the 15,108 participants who began our study, 13,751 finished Part 1. Other than the discussion of dropout rates, our analysis focuses on these participants or a subset of these participants. 8,565 returned for Part 2 within 3 days of receiving our invitation; 8,143 of them finished Part 2. When discussing metrics from Part 2, we focus on these participants.

Of the participants, 51.5% reported being male and 47.4% reported being female. Participants' mean age was 29.3 years (median 26). These did not vary significantly

[4]http://www.openwall.com/wordlists/.

Table II. Summary of Password Attributes and Creation Failure on the First Attempt

Condition	Participants	Length (median)	Upper (median)	Lower (median)	Digit (median)	Sym. (median)	Fail (%)	Length (%)	Class (%)	Dict. (%)	2word (%)
comp8	1996	10	1	5	2	1	58.0	6.5	26.3	39.0	–
basic12	1693	13	0	10	3	0	40.6	38.2	–	18.5*	–
basic16	1757	17	0	14	3	0	52.6	50.4	–	6.3*	–
basic20	1715	21	0	18	3	0	59.9	57.3	–	4.3*	–
3class12	1653	13	1	8	3	1	44.5	38.2	9.5	23.4*	–
3class16	1625	17	1	11	3	1	52.2	47.2	10.0	9.7*	–
2word12	1659	14	0	11	2	0	54.5	30.4	9.9	6.5*	45.4
2word16	1653	18	0	14	2	1	59.8	44.8	9.6	2.6*	45.1

A password can fail multiple ways. We omit failure from blank fields and confirmation mismatch. *Dict* shows the percent of comp8 participants who failed the dictionary check on their first attempt. It also shows the percentage of final passwords in other conditions that would have failed the dictionary check.

between conditions. Looking at user-agent strings, only 1.5% of participants appeared to be using mobile devices.

4.3. Security Results

We next describe our findings for password strength, the calculation of which we discussed in Section 3.2. Table II has descriptive statistics for how users created their passwords. For example, participants typically avoided uppercase letters or symbols in their passwords, but often included digits.

Figure 1 shows the percentage of passwords cracked in each condition as additional guesses are made. Table I shows the percentages of passwords guessed in each condition after 10^6 and 10^{14} guesses, with pairwise significant differences in Table III.

Conditions basic16 and basic20 performed relatively well after 10^{14} guesses. However, they contained a number of easily guessed passwords and therefore performed poorly against an attacker limited to 10^6 guesses. The comp8 condition was relatively strong against an attacker limited to 10^6 guesses but performed poorly against an attacker capable of making 10^{14} guesses. This is consistent with the findings of our prior work [Kelley et al. 2012].

Several conditions combining longer length and character-class requirements, such as 3class12 and 2word16, performed well across a range of guess numbers. Condition 3class16 was strong after both 10^6 and 10^{14} guesses. The 2word and 3class conditions were stronger than their basic counterparts. 3class12 was similar to comp8 until around 10^{10} guesses and is stronger after.

It is interesting to note the disparity between 2word12 and 2word16. Adding the 2word requirement improves basic16 more than basic12. Examining participants who finished Part 2 shows that 2word16 participants created passwords with three or more words 31.8% of the time, almost twice as often as 2word12 participants. Examples of these passwords from 2word16 include *bite-the-wax-tadpole* and *kill the vampire in your life*. The 2word approach seems more effective with a length-16 requirement, at least in part because this leads to more participants creating passphrases.

To understand how participants perceived the strength of their study passwords, we asked whether they agreed with the statement, "If my main email provider had the same password requirements as used in this study, my email account would be more secure." Agreement ranged from 59.8% for 3class16 and 59.7% for comp8 to 35.2% for basic12. Participants in comp8 were more likely to agree than in any other condition except 3class16. This suggests that user perception of password strength does not align with our strength analysis. We suspect that participants expected the comp8 requirements to lead to strong passwords because it is similar to a traditional "strong" policy.

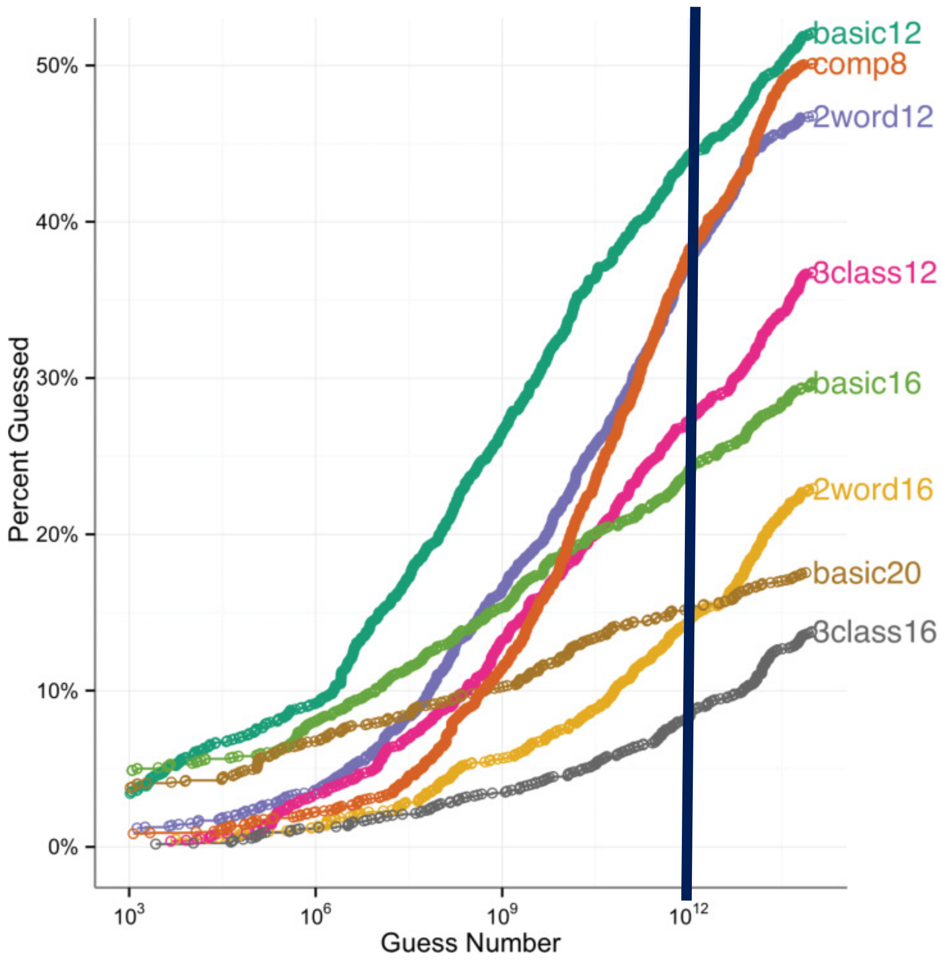

Fig. 1. The percentage of passwords cracked in each condition by the number of guesses made, in log scale. Our cutoff for guess numbers was 10^{14}. Table I shows significant differences in cracking rates between conditions.

4.4. Usability Results

In this section, we examine dropout rates, as well as password storage, creation, and recall. Overall, we found that most conditions are significantly more usable than comp8 on a number of metrics.

4.4.1. Study Dropout. We consider higher dropout rates to indicate increased confusion, boredom, or frustration. We interpret statistically significant differences in dropout rates between conditions to be caused by differing usability between conditions. Among 15,108 participants who began the study, 91.0% finished Part 1. Part 1 completion varied significantly by condition (χ^2_7=246.60, p<.001), ranging from 83.0% for comp8 to 94.5% for basic12. Participants in comp8 were significantly less likely to finish Part 1 than those in any other condition (HC χ^2, p<.001). Participants in 3class16 (90.5%) were significantly less likely to finish Part 1 than those in basic12 (94.5%) or basic16 (93.9%) (HC χ^2, p<.004). Of those participants who finished Part 1, 62.3% returned within 3 days of being invited back; this did not vary significantly by condition (χ^2_7=7.69,

Table III. Significant Differences in the Probability of Passwords Cracked after 10^6 and 10^{14} Guesses, Representing More and Less Resource-Constrained Attackers

Cracked passwords after 10^6 guesses
Omnibus χ^2_7=270.784, p<.001

cond 1	%	cond 2	%	p-value
basic12	9.1%	basic20	5.6%	.001
		2word12	3.4%	<.001
		3class12	3.2%	<.001
		comp8	2.2%	<.001
		3class16	1.2%	<.001
		2word16	1.1%	<.001
basic16	7.9%	2word12	3.4%	<.001
		3class12	3.2%	<.001
		comp8	2.2%	<.001
		3class16	1.2%	<.001
		2word16	1.1%	<.001
basic20	5.6%	2word12	3.4%	.025
		3class12	3.2%	.008
		comp8	2.2%	<.001
		3class16	1.2%	<.001
		2word16	1.1%	<.001
2word12	3.4%	3class16	1.2%	<.001
		2word16	1.1%	<.001
3class12	3.2%	3class16	1.2%	<.001
		2word16	1.1%	<.001

Cracked passwords after 10^{14} guesses
Omnibus χ^2_7=1238.038, p<.001

cond 1	%	cond 2	%	p-value
basic12	52.0%	2word12	46.6%	.007
		3class12	36.8%	<.001
		basic16	29.7%	<.001
		2word16	22.9%	<.001
		basic20	16.4%	<.001
		3class16	13.8%	<.001
comp8	50.1%	3class12	36.8%	<.001
		basic16	29.7%	<.001
		2word16	22.9%	<.001
		basic20	16.4%	<.001
		3class16	13.8%	<.001
2word12	46.6%	3class12	36.8%	<.001
		basic16	29.7%	<.001
		2word16	22.9%	<.001
		basic20	16.4%	<.001
		3class16	13.8%	<.001
3class12	36.8%	basic16	29.7%	<.001
		2word16	22.9%	<.001
		basic20	16.4%	<.001
		3class16	13.8%	<.001
basic16	29.7%	2word16	22.9%	<.001
		basic20	16.4%	<.001
		3class16	13.8%	<.001
2word16	22.9%	basic20	16.4%	<.001
		3class16	13.8%	<.001

Figure 1 illustrates these guess numbers along a curve. In both tables, the more secure condition is in the cond 2 column.

p=0.361). Of those who returned for Part 2, 95.1% completed Part 2 within 3 days of being invited back; this also did not vary significantly by condition (χ^2_7=4.15, p=0.762).

4.4.2. Password Storage. Our analysis of password storage looked at participants who finished Part 2, because we asked participants about their storage behavior only in the Part 2 survey. To analyze storage, we classify participants into two groups: storage and nonstorage participants. To be considered a nonstorage participant, the participant must tell us the password was not stored in two separate questions in the Part 2 survey. Further, the participant must not be detected pasting or using browser autocomplete in Part 2 Recall, except after returning via the password-reminder link.

Of the participants, 52.6% were *storage* participants. This ranged from 45.4% for basic12 to 60.2% for 3class16; Table IV shows significant pairwise differences. 3class16 had a significantly higher storage rate than every other condition except comp8 and 3class12. Password storage rates were highest in conditions that required three or four character classes, and lowestin the basic conditions.

4.4.3. Password Creation. We interpret taking more password-creation attempts as being less usable. Participants took an average of 1.9 attempts to create a password. Table IV shows significant pairwise differences. comp8 took the most attempts (mean=2.4) and basic12 took the fewest (mean=1.5). We asked participants whether they agreed

Table IV. Significant Differences in Study 1 Password-Creation Usability, with the Significantly More Usable Condition in the Cond 2 Column

Password creation attempts
Omnibus KW χ^2_7=394.337, p<.001

cond 1	mean	cond 2	mean	p-value
comp8	2.4	basic20	1.9	.011
		3class16	1.9	<.001
		2word12	1.9	<.001
		basic16	1.8	<.001
		3class12	1.7	<.001
		basic12	1.5	<.001
2word16	2.1	3class16	1.9	<.001
		2word12	1.9	<.001
		basic16	1.8	<.001
		3class12	1.7	<.001
		basic12	1.5	<.001
basic20	1.9	3class16	1.9	.011
		2word12	1.9	.023
		basic16	1.8	<.001
		3class12	1.7	<.001
		basic12	1.5	<.001
3class16	1.9	3class12	1.7	<.001
		basic12	1.5	<.001
2word12	1.9	3class12	1.7	<.001
		basic12	1.5	<.001
basic16	1.8	3class12	1.7	<.001
		basic12	1.5	<.001
3class12	1.7	basic12	1.5	.006

Password entry time (s)
Omnibus KW χ^2_7=71.58, p<.001

cond 1	median	cond 2	median	p-value
3class16	16.2	comp8	13.2	.001
		2word12	13.1	<.001
		basic12	11.6	<.001
basic20	15.3	comp8	13.2	<.001
		2word12	13.1	<.001
		basic12	11.6	<.001
3class12	14.8	basic12	11.6	.001
2word16	14.6	2word12	13.1	.012
		basic12	11.6	<.001
basic16	13.7	basic12	11.6	.003

Password storage
Omnibus χ^2_7=61.87, p<.001

cond 1	%	cond 2	%	p-value
3classc16	60.2	2word12	51.4	.002
		2word16	51.3	.002
		basic20	50.0	<.001
		basic16	49.9	<.001
		basic12	45.4	<.001
comp8	56.9	basic20	50.0	.029
		basic16	49.9	.02
		basic12	45.4	<.001
3class12	54.9	basic12	45.4	<.001

Agree password creation difficult
Omnibus χ^2_7=239.44, p<.001

cond 1	%	cond 2	%	p-value
3class16	40.3%	basic20	35.2%	.019
		2word16	34.7%	.007
		comp8	32.8%	<.001
		basic16	28.5%	<.001
		3class12	26.0%	<.001
		2word12	21.9%	<.001
		basic12	15.2%	<.001
basic20	35.2%	basic16	28.5%	<.001
		3class12	26.0%	<.001
		2word12	21.9%	<.001
		basic12	15.2%	<.001
2word16	34.7%	basic16	28.5%	<.001
		3class12	26.0%	<.001
		2word12	21.9%	<.001
		basic12	15.2%	<.001
comp8	32.8%	basic16	28.5%	.027
		3class12	26.0%	<.001
		2word12	21.9%	<.001
		basic12	15.2%	<.001
basic16	28.5%	2word12	21.9%	<.001
		basic12	15.2%	<.001
3class12	26.0%	2word12	21.9%	.032
		basic12	15.2%	<.001
2word12	21.9%	basic12	15.2%	<.001

Agree remembering password difficult
Omnibus χ^2_7=83.89, p<.001

cond 1	%	cond 2	%	p-value
3class16	42.9	3class12	35.3	.012
		basic20	32.9	<.001
		2word12	31.0	<.001
		basic16	30.1	<.001
		basic12	27.4	<.001
comp8	39.3	basic20	32.9	.035
		2word12	31.0	.001
		basic16	30.1	<.001
		basic12	27.4	<.001
2word16	36.8	basic16	30.1	.03
		basic12	27.4	<.001
3class12	35.3	basic12	27.4	.002

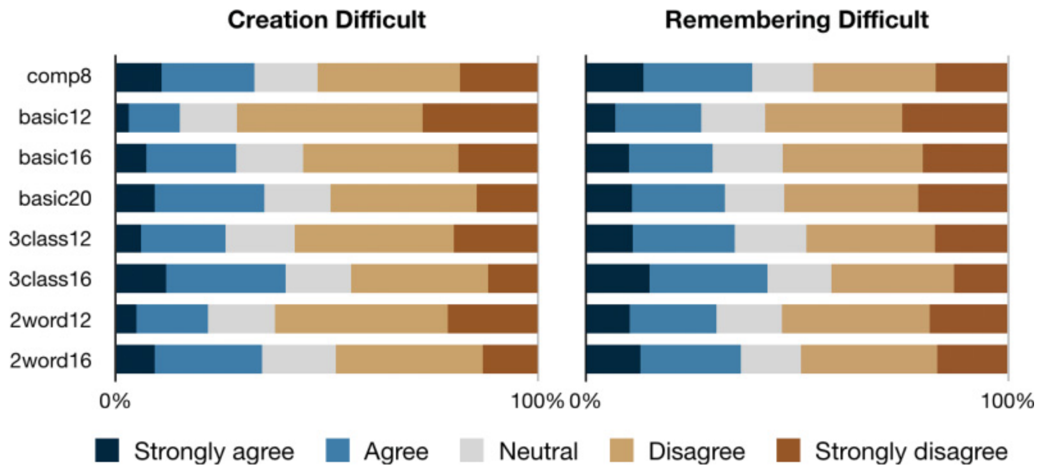

Fig. 2. Participant agreement with “Creating a password that meets the requirements given in this study was difficult” and “Remembering the password I used for this study was difficult.” Significant differences are in Table IV.

with the statement, “Creating a password that meets the requirements given in this study was difficult.” Figure 2 shows responses and Table IV shows significant differences. Agreement ranged from 15.2% for basic12 to 40.3% for 3class16.

4.4.4. Creation Failure. For a better understanding of password-creation failures, we looked at participants’ first failed attempt. Table II shows these failures. Many participants failed to meet length requirements, with 57.3% of participants in basic20 using less than 20 characters. 26.3% of participants in comp8 used too few character classes, compared to between 9% and 10% of participants in other conditions that required nonletter characters. This suggests that participants struggle more to create a password with four classes compared to three. The largest source of failure in comp8 was the dictionary check. Only comp8 had a dictionary check, so we looked at how many passwords in other conditions would have been prevented by that check. These numbers are included in the same column in Table II with an asterisk. This was 23.4% of passwords in 3class12, 18.5% in basic12, and less than 10% in any other condition. The dictionary check strips away nonletter characters and checks the remaining letters against a dictionary. We speculate that the longer conditions were less likely to fail the dictionary check because longer passwords tend to have more letters than a single dictionary word.

4.4.5. Part 1 Recall. Participants recalled their passwords after filling out a brief survey in Part 1. 93.5% of participants correctly entered their password on the first attempt and this varied by condition (χ^2_7=27.241, p<0). Participants in basic12 (95.7%) were significantly more likely to enter their passwords correctly on the first try than those in 2word16 (92.9%), 3class12 (93.1%), 3class16 (92.1%), or basic20 (92.5%) (HC FET, p<.038).

4.4.6. Part 2 Recall. In Part 2 Recall, participants could use a password reminder to display their password. Of the participants, 15.5% used this feature, and this did not vary significantly by condition (χ^2_7=8.31, p=0.306). Among no-storage participants, 21.4% used the reminder, and this also did not vary by condition (χ^2_7=7.72, p=0.358). 80.1% of participants successfully entered their password in five attempts without using the reminder, and this did not vary significantly by condition (χ^2_7=7.75, p=0.356).

Table V. Substrings in at Least 1% of Passwords

Substring	Using	Cracked | Using	Cracked | ¬Using	*p*-value
1234	4.9%	69.9%	32.3%	<.001
password	3.0%	54.0%	33.5%	<.001
123456789	1.7%	79.0%	33.3%	<.001
turk	1.5%	45.4%	34.0%	.004
char	1.1%	44.0%	34.0%	.048
love	1.9%	34.1%	34.1%	1.000
2013	1.6%	31.4%	34.2%	1.000
this	1.6%	31.8%	34.2%	1.000

The second column shows the percent of passwords containing the substring. The next two show percentages of passwords cracked containing and not containing it. The fifth shows a χ^2 test on the difference. The presence of “2013” likely results from the study being conducted in that year.

These participants took an average of 1.3 tries, and this also did not vary significantly by condition (χ^2_7=12.96, p=0.073).

As a measure of usability, we looked at how long participants spent entering their passwords on their first successful attempt. We looked only at no-storage participants who did not use the reminder. Median times varied from basic12 (11.6 seconds) to 3class16 (16.2 seconds), with significant differences in Table IV. Overall, participants in 3class16 took the most time to enter their passwords successfully. Participants in basic12 and 2word12 took the least time to enter their passwords on the first successful attempt, despite these passwords being longer than comp8 passwords.

To measure subjective user difficulty, we asked participants whether they agreed with the statement, “Remembering the password I used for this study was difficult.” The least difficult condition for recall was basic12 (27.4%), and the most difficult were comp8 (39.3%) and 3class16 (42.9%). Figure 2 depicts the results, and Table IV shows significant differences.

4.5. Password Patterns

We next look at themes and patterns that emerge in our study passwords for a better understanding of how users create passwords.

4.5.1. Common Substrings. We identified the most common substrings in the study's passwords. We first made a list of all substrings of 4 to 12 characters present in at least 1% of the passwords. We removed any substrings that did not exist in at least 1% of study passwords without already being part of another, longer substring on the list. For example, we removed “sword,” which was almost always present as part of “password.” This left the eight substrings in Table V. Overall, 1,944 passwords (14.1%) contain at least one of the eight substrings. We looked at password-cracking rates for passwords with and without each substring. Table V shows the five substrings such that passwords containing any of them were significantly more likely to be cracked. This finding suggests future research on proactively checking prospective passwords and rejecting any password that contains a substring associated with weak passwords.

4.5.2. Meeting the Comp8 Requirements. Of the passwords, 29.0% in comp8 fulfilled the symbol requirement only by placing “!” at the end of the password, and 57.7% in comp8 used an uppercase letter as their first character and used no other uppercase letter. Passwords doing either of these were significantly more likely to be cracked (62.9% to 26.3% cracked) (χ^2_1=240.5276, p<.001). This suggests comp8 can be far more effective when its requirements are not met in minimal ways.

4.5.3. Going Beyond the Requirements. Table II shows that participants often exceeded minimum length and character-class requirements. Each condition has a median length above its minimum, and all conditions have a median of at least two digits. 66.4% of participants exceeded their minimum required length, ranging from 59.3% of participants in basic12 to 76.8% in comp8. Perhaps not surprisingly, passwords that exceeded the minimum length requirements were less likely to be cracked (28.2% to 45.9%) (χ^2_1=426.8069, p<.001).

We also looked at exceeding the minimum number of character classes, omitting comp8 because it already required all four character classes. Of the non-comp8 participants 62.4% used more than the minimum number of character classes. Of the participants, 38.5% in 2word16 and 38.5% in 2word12 used at least three character classes. Over two thirds of passwords in each of the basic and 3class conditions exceeded their minima. Passwords exceeding the minimum were significantly less likely to be cracked, 24.9% to 42.2%, (χ^2_1=382.291, p<.001).

4.5.4. Character Distribution in 3Class12. Participants often responded to the three-character-class requirement of 3class12 by placing characters other than lowercase letters at the beginning or end of their passwords. In fact, fewer than 2% of passwords in 3class12—33 of 1,653—began and ended with lowercase letters. Those 33 passwords were particularly difficult to guess. After 10^6 guesses, none were guessed. After 10^{14} guesses, only one of them was guessed (“family-4ever”). This suggests that encouraging or requiring users to distribute their different character classes more evenly might make character-class requirements more effective.

4.5.5. Semantic Analysis. For a better understanding of the semantic content of the passwords, we looked at 100 random passwords per condition from participants who finished Part 2. Participants who included words were more likely to place nonletter characters between words rather than within them. Names, dates, and sequences of characters such as “1234” and “qwerty” were common. We saw a number of study-related words, and references to animals, love, and pop culture. Consistent with those themes, looking at all passwords and ignoring case, 42 passwords contained “monkey” and 294 passwords contained “love.” Future research might explore encouraging participants to choose words from a wider range of themes and to add special characters within words.

5. USING FINDINGS FROM STUDY 1 TO CREATE STUDY 2 CONDITIONS

Looking at Table I, both 3class12 and 2word16 stand out. They performed significantly better than comp8 on several usability metrics, were more secure after 10^{14} guesses, and did not perform worse than comp8 on any metric. Comparing 3class12 and 2word16, passwords in 2word16 were more secure. However, passwords in 3class12 were easier to create, and still significantly more secure than those in comp8. 3class12 is also more similar to policies commonly found in the wild, so participants may be more comfortable with conditions based on it. Therefore, Study 2 used 3class12 as its baseline condition. Exploring conditions based on 2word16 remains promising future work.

Section 4.5 showed that a small set of substrings were markers of passwords being more likely to be cracked. For example, 79.0% of passwords containing “123456789” were cracked, compared to 33.3% without it. Therefore, we introduced the *blacklist* requirement, which requires that passwords not contain any blacklisted substrings. We hypothesized that preventing common substrings would prevent some easily guessed passwords. While users may not be familiar with the concept of a substring blacklist, we hoped that familiarity with common dictionary checks would help them grasp it. In practice, checking a short substring blacklist would be more feasible to conduct client-side than a traditional dictionary check. The blacklist we used in Study 2 is below. Of Study 1 passwords, 69.2% would have passed this requirement, ranging from 57.4% of basic20 to 83.0% of comp8.

As discussed in Section 4.5, fewer than 2% of 3class12 passwords began and ended with lowercase letters, but those that did were especially difficult to guess. We call the requirement that passwords begin and end with a lowercase letter the *pattern* requirement. We hypothesized it would lead users to distribute nonlowercase letters more evenly throughout their password.

2class12, 3class12—These conditions required 12 characters and two and three character classes. We expected that adding more requirements directly to 3class12 would result in conditions less usable than 3class12 itself. Therefore, we created 2class12, which is similar to 3class12 but requires one fewer character-class. We created the other conditions by adding requirements to 2class12.

2class16—3class16 passwords were difficult to create and recall but were also difficult to guess. We observed that password length was often more usable than character complexity. We hoped that reducing the character-class requirements could make 3class16 more usable.

2list12, 2s-list12—These conditions combined the blacklist requirement with the requirements of 2class12. Participants in 2list12 saw an explicit list of blacklisted substrings, and those in 2s-list12 were simply told "Do not include words commonly found in passwords (e.g. *password*), keyboard patterns (e.g., *qazx*), or other common patterns (e.g. *5678*)." We used the following blacklist.

—123!, amazon, character, monkey, number, survey, this, turk
—Any year between 1950 and 2049
—The same character four or more times in a row
—Any four consecutive characters from password
—Any four sequential digits (e.g., 5678)
—Any four sequential letters in the alphabet (e.g., wxyz)
—Any four consecutive characters on the keyboard (e.g., wsxc)

2pattern12—This combined 2class12 with the pattern requirement, that the password start and end with a lowercase letter. We hoped that this would lead to a more even distribution of special characters within passwords, making passwords more difficult to guess.

2list-patt12, 2s-list-patt12—These conditions combined the blacklist and pattern requirements.

6. STUDY 2 FINDINGS

Section 6.1 presents participant demographics in order to provide context for the rest of the results of our second study. We present security results in Section 6.2 and usability results in Section 6.3. Section 6.4 discuss patterns we observed in cracked patterns. Table VI summarizes the findings for this second study.

6.1. Participants

We collected data from December 2013 to January 2014. 9,707 participants began the study, 8,740 finished Part 1, and 5,111 returned for Part 2 within 3 days of being notified. Table VII shows the number of participants per condition. Participant age did not vary significantly by condition (χ^2_7=10.069, p=0.185) (mean=30.59, standard deviation = 10.45, median = 28). Reported gender did not vary significantly either (χ^2_7=4.821, p=0.682) (47.9% male, 51.2% female).

Table VI. A Summary of Findings for Study 2

Condition	Part 1 completion (%)	Password storage (%)	Mean creation attempts	Agree creation difficult (%)	Part 2 recall attempts	Part 2 entry time (s)	Agree remembering difficult (%)	Cracked@10^6 (%)	Cracked@10^{14} (%)
3class12	**92.0**	**52.7**	**1.6**	**24.1**	1.8	**15.28**	**36.0**	**2.9**	**40.1**
2class12	93.3	50.8	1.6	25.1	1.7	15.09	35.4	2.1	37.6
2class16	90.4	56.7	1.8	40.1	1.7	18.60	38.5	1.4	14.2
2list12	91.9	59.6	1.8	32.8	1.7	14.97	35.7	0.8	28.5
2s-list12	90.5	56.5	1.9	27.4	1.8	15.64	32.6	1.2	31.3
2pattern12	88.7	61.7	2.4	46.8	1.7	19.00	47.4	0.7	17.1
2list-patt12	87.4	64.0	2.4	50.0	1.7	18.66	49.1	0.2	11.8
2s-list-patt12	86.0	67.5	2.6	50.2	1.7	19.38	49.0	0.0	9.9

Each condition is compared to 3class12. Light blue indicates being statistically significantly better than 3class12, and dark red indicates being worse. No shading indicates no statistically significant difference.

Table VII. Password Attributes and Creation Failure on the First Try

Condition	Participants	Len. med	Up. med	Low. med	Digit med	Sym. med	Fail (%)	Length (%)	Class (%)	Blacklist (%)	Pattern (%)
3class12	1,121	13	1	8	3	1	43.0	37.6	8.4	35.6*	98.0*
2class12	1,131	13	1	8	3	1	41.2	37.2	1.7	32.8*	97.3*
2class16	1,096	17	1	12	3	1	51.9	47.5	2.0	38.3*	96.2*
2list12	1,113	13	1	8	3	1	46.7	29.4	1.2	16.3	96.0*
2s-list12	1,099	13	1	8	3	1	52.3	33.5	1.7	19.6	97.0*
2pattern12	1,076	14	1	9	2	1	70.5	35.2	1.3	28.5*	59.1
2list-patt12	1,059	14	1	9	2	1	69.1	27.9	1.8	13.3	55.9
2s-list-patt12	1,045	14	1	9	2	1	76.7	35.3	2.6	21.8	58.1

Length and character counts are medians. A password can fail in multiple ways. We omit failure from blank fields and confirmation mismatch. “Blacklist” and “Pattern” show percents of participants who failed those checks for conditions with those checks. For other conditions (marked with *), they show the percentage of final passwords that would have failed.

6.2. Security Results

We next compare password strength across conditions. Figure 3 depicts the proportion of passwords in each condition that were guessed as the number of guesses increased. Table VIII shows significant differences in guessing after 10^6 and 10^{14} guesses. After 10^{14} guesses, 3class12 and 2class12 did not differ significantly, but they performed significantly worse than any other condition. 2list12 and 2s-list12 performed significantly worse than 2class16 or the conditions using the pattern requirement. The strongest conditions were 2list-patt12, and 2s-list-patt12.

Table VIII shows differences in cracking after 10^6 guesses. This smaller number of guesses simulates an online attack. 3class12 and 2class12 performed worse than the conditions using both a blacklist and the pattern requirement, though fewer than 3% of passwords in any condition were guessed.

6.3. Usability Results

Overall, 2class12 and 3class12 had similar usability metrics and were more usable than the other conditions. The blacklist check made password creation more difficult

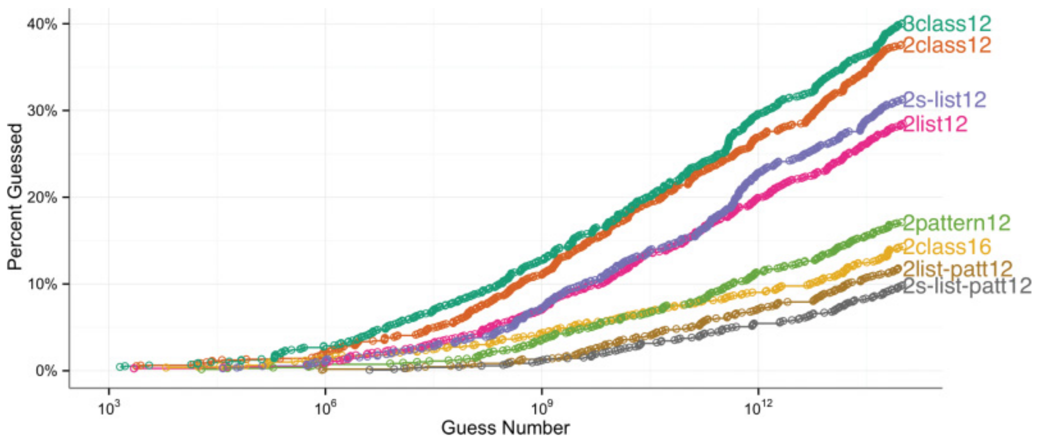

Fig. 3. The percentage of passwords cracked in each condition by the number of guesses made in log scale. Our cutoff for guess numbers was 10^{14}. Table VIII shows significant differences in cracking rates between conditions.

but did not affect recall. The pattern requirement negatively affected creation and recall usability.

6.3.1. Study Dropout. Table VIII shows significant differences in Part 1 dropout rates (starting but not finishing Part 1). Participants in the pattern conditions were the most likely to drop out, which may suggest they were more frustrated. There was no significant difference in rates of returning for Part 2 (χ^2_7=5.826, p=0.56) or completing Part 2 (χ^2_7=5.97, p=0.543).

6.3.2. Password Storage. Of the Part 2 participants, 58.5% stored their password. Table VIII shows significant differences across conditions. Participants in the pattern conditions were the most likely to store their passwords, which may indicate actual or expected recall difficulty.

6.3.3. Password Creation. Looking at password creation, Table VIII shows significant differences in password-creation attempts. Participants in the pattern conditions took significantly more attempts than in any nonpattern condition. Fewer than one third of participants in pattern conditions successfully created a password on the first try. In the other conditions, this ranged from 48% for 2s-list12 to 59% for 2class12.

Over a third of participants in 2pattern12 took three or more attempts to create a password. A subset of participants, 7.2%, took five or more attempts. One participant, for example, took 10 attempts to create a password that matched the requirements of 2pattern12. All except the final, successful attempt of this participant ended with either a digit or an exclamation point.

As we did in the first study, to learn about perceived password-creation difficulty, we asked participants whether they agreed with the statements "Creating a password that meets the requirements given in this study was annoying" and "Creating a password that meets the requirements given in this study was difficult." Figure 4 depicts responses, and Table IX shows pairwise differences. Password creation under the the pattern conditions was more annoying and usually more difficult than under the nonpattern conditions.

Anecdotal evidence from our previous research suggested that users were fond of incorporating the term "monkey" into their passwords. To examine this more scientifically, we detected passwords with text matching or similar to "monkey." After these

Table VIII. Significant Differences between Conditions in Study 2

Cracked after 10^6 guesses
Omnibus χ^2_7=60.451, p<.001

cond 1	%	cond 2	%	p-value
3class12	2.9%	2list12	0.8%	.008
		2pattern12	0.7%	.004
		2list-patt12	0.2%	<.001
		2s-list-patt12	0.0%	<.001
2class12	2.1%	2list-patt12	0.2%	<.001
		2s-list-patt12	0.0%	<.001
2class16	1.4%	2list-patt12	0.2%	.046
		2s-list-patt12	0.0%	.001
2s-list12	1.2%	2s-list-patt12	0.0%	.005

Cracked after 10^{14} guesses
Omnibus χ^2_7=602.357, p<.001

cond 1	%	cond 2	%	p-value
3class12	40.1%	2s-list12	31.3%	<.001
		2list12	28.5%	<.001
		2pattern12	17.1%	<.001
		2class16	14.2%	<.001
		2list-patt12	11.8%	<.001
		2s-list-patt12	9.9%	<.001
2class12	37.6%	2s-list12	31.3%	.015
		2list12	28.5%	<.001
		2pattern12	17.1%	<.001
		2class16	14.2%	<.001
		2list-patt12	11.8%	<.001
		2s-list-patt12	9.9%	<.001
2s-list12	31.3%	2pattern12	17.1%	<.001
		2class16	14.2%	<.001
		2list-patt12	11.8%	<.001
		2s-list-patt12	9.9%	<.001
2list12	28.5%	2pattern12	17.1%	<.001
		2class16	14.2%	<.001
		2list-patt12	11.8%	<.001
		2s-list-patt12	9.9%	<.001
2pattern12	17.1%	2list-patt12	11.8%	.004
		2s-list-patt12	9.9%	<.001
2class16	14.2%	2s-list-patt12	9.9%	.015

Part 1 dropout rate
Omnibus χ^2_7=58.579, p<.001

cond 1	%	cond 2	%	p-value
2s-list-patt12	14.0%	2class16	9.6%	.021
		2s-list12	9.5%	<.013
		2list2	8.1%	<.001
		3class12	8.0%	<.001
		2class12	6.7%	<.001
2list-patt12	12.6%	2list12	8.1%	.008
		3class12	8.0%	.005
		2class12	6.7%	<.001
2pattern12	11.3%	2class12	6.7%	.002

Proportion of storage participants
Omnibus χ^2_7=57.391, p<.001

cond 1	%	cond 2	%	p-value
2s-list-patt12	67.5%	2class16	56.7%	.002
		2s-list12	56.5%	.002
		3class12	52.7%	<.001
		2class12	50.8%	<.001
2list-patt12	64.0%	3class12	52.7%	.002
		2class12	50.8%	<.001
2pattern12	61.7%	3class12	52.7%	.031
		2class12	50.8%	.002
2list12	59.6%	2class12	50.8%	.032

Password creation attempts
Omnibus KW χ^2_7=795.632, p<.001

cond 1	count	cond 2	count	p-value
2s-list-patt12	2.6	2list-patt12	2.4	.001
		2pattern12	2.4	.001
		2s-list12	1.9	<.001
		2class16	1.8	<.001
		2list12	1.8	<.001
		3class12	1.6	<.001
		2class12	1.6	<.001
2list-patt12	2.4	2list-patt12	1.9	<.001
		2class16	1.8	<.001
		2list12	1.8	<.001
		3class12	1.6	<.001
		2class12	1.6	<.001
2pattern12	2.4	2s-list12	1.9	<.001
		2class16	1.8	<.001
		2list12	1.8	<.001
		3class12	1.6	<.001
		2class12	1.6	<.001
2s-list12	1.9	3class12	1.6	<.001
		2class12	1.6	<.001
2class16	1.8	3class12	1.6	<.001
		2class12	1.6	<.001
2list12	1.8	3class12	1.6	.036
		2class12	1.6	<.001

The first column shows differences in proportions of passwords cracked after 10^6 and 10^{14} guesses, with the significantly more secure condition on the right. The next column shows significant usability differences. These include differences in the Part 1 drop rate, Storage rate for Part 2 participants, and the number of attempts needed to create a satisfactory password. In each case, the more usable condition is in the Cond 2 column.

Table IX. Significant Usability Differences

Agreement with creation difficult

Omnibus χ^2_7=405.645, p<.001

cond 1	%	cond 2	%	p-value
2s-list-patt12	50.2%	2class16	40.1%	<.001
		2list12	32.8%	<.001
		2s-list12	27.4%	<.001
		2class12	25.1%	<.001
		3class12	24.1%	<.001
2list-patt12	50.0%	2class16	40.1%	<.001
		2list12	32.8%	<.001
		22s-list12	27.4%	<.001
		2class12	25.1%	<.001
		3class12	24.1%	<.001
2pattern12	46.8%	2class16	40.1%	.015
		2list12	32.8%	<.001
		2s-list12	27.4%	<.001
		2class12	25.1%	<.001
		3class12	24.1%	<.001
2class16	40.1%	2list12	32.8%	.003
		2s-list12	27.4%	<.001
		2class12	25.1%	<.001
		3class12	24.1%	<.001
2list12	32.8%	2s-list12	27.4%	.044
		2class12	25.1%	<.001
		3class12	24.1%	<.001

Agreement with recall being difficult

Omnibus χ^2_7=85.906, p<.001

cond 1	%	cond 2	%	p-value
2list-patt12	49.1%	2class16	38.5%	.003
		3class12	36.0%	<.001
		2list12	35.7%	<.001
		2class12	35.4%	<.001
		2s-list12	32.6%	<.001
2s-list-patt12	49.0%	2class16	38.5%	.002
		3class12	36.0%	<.001
		2list12	35.7%	<.001
		2class12	35.4%	<.001
		2s-list12	32.6%	<.001
2pattern12	47.4%	2class16	38.5%	.019
		3class12	36.0%	<.001
		2list12	35.7%	<.001
		2class12	35.4%	<.001
		2s-list12	32.6%	<.001

Time for participants successful on the first try in Part 1

Omnibus KW χ^2_7=117.625, p<.001

cond 1	median	cond 2	median	p-value
2class16	8.9	2list-patt12	8.4	.039
		2list12	8.0	<.001
		3class12	7.7	<.001
		2s-list12	7.7	<.001
		2class12	7.2	<.001
2pattern12	8.8	2list12	8.0	.003
		3class12	7.7	<.001
		2s-list12	7.7	<.001
		2class12	7.2	<.001
2s-list-patt12	8.7	2list12	8.0	.033
		3class12	7.7	.002
		2s-list12	7.7	.006
		2class12	7.2	<.001
2list-patt12	8.4	2class12	7.2	<.001
2list12	8.0	2class12	7.2	.001
3class12	7.7	2class12	7.2	.021
2s-list12	7.7	2class12	7.2	.01

Agreement with creation annoying

Omnibus χ^2_7=303.957, p<.001

cond 1	%	cond 2	%	p-value
2list-patt12	77.3%	2class16	70.0%	.001
		2list12	61.4%	<.001
		2s-list12	57.9%	<.001
		3class12	57.3%	<.001
		2class12	54.0%	<.001
2s-list-patt2	76.0%	2class16	70.0%	.019
		2list12	61.4%	<.001
		2s-list12	57.9%	<.001
		3class12	57.3%	<.001
		2class12	54.0%	<.001
2pattern12	74.7%	2list12	61.4%	<.001
		2s-list2	57.9%	<.001
		3class12	57.3%	<.001
		2class12	54.0%	<.001
2class16	70.0%	2list12	61.4%	<.001
		2s-list12	57.9%	<.001
		3class12	57.3%	<.001
		2class12	54.0%	<.001
2list12	61.4%	2class12	54.0%	.005

The more usable conditions are in the Cond 2 column. The first column shows agreement with password creation being annoying and difficult. The second shows Part 1 Recall timing and agreement with recall being difficult.

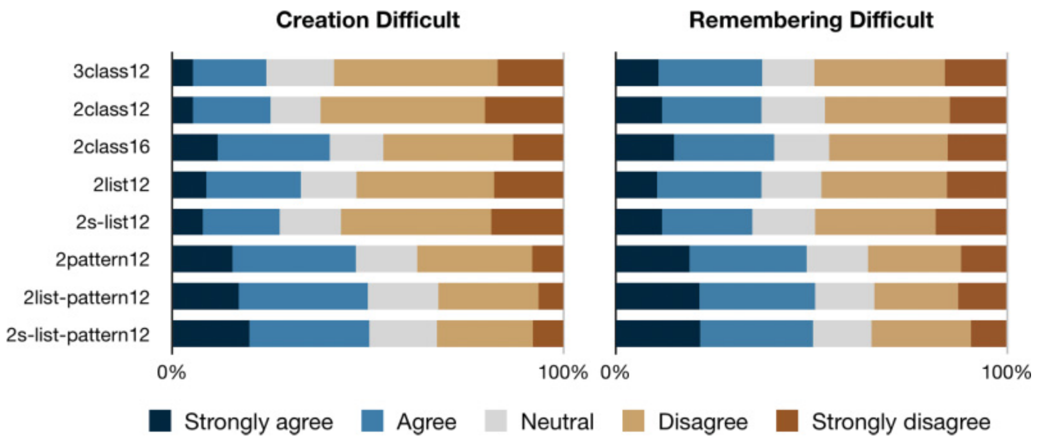

Fig. 4. Participant agreement with "Creating a password that meets the requirements given in this study was difficult" and "Remembering the password I used for this study was difficult.".

participants finished Part 1, we asked, "We couldn't help but notice that you have a Monkey-ish word in your password. Please tell us why you included [text] in your password." We detected 17 passwords with a monkey-ish phrase (0.2%). Participants reported liking monkeys, finding them cute, and having "monkey" be a nickname or pet name. This is further anecdotal evidence that users often make passwords related to things they like.

6.3.4. Creation Failure. Table VII shows types of password-creation failures on the first attempt. Participants struggled to meet the pattern requirement. Over two thirds of participants with this requirement failed on their first attempt, and over half of these failures were due to the pattern requirement itself. The blacklist requirement appears to have prevented fewer passwords than the dictionary check of comp8.

6.3.5. Part 1 Recall. In Part 1 recall, 93.2% of participants entered their passwords correctly on the first attempt. This was not significantly different between conditions (χ^2_7=5.101, p=.648). Table IX shows significant differences in password-entry time for participants who correctly entered their password on the first try. Passwords in 2class16 took longest to enter, followed by 2pattern12. 2class12 took significantly less time than any other condition. However, the effect size is small. Median times ranged from 7.2 seconds (2class12) to 8.9 seconds (2class16).

6.3.6. Part 2 Recall. Of the participants, 5,111 returned and finished Part 2 within 3 days of being invited back; 14.9% of them used the reminder and this did not vary significantly by condition (χ^2_7=6.833, p=0.446); and 80.0% of participants entered their password correctly within five attempts without the reminder, and this also did not vary by condition (χ^2_7=5.401, p=0.611). Among these success participants, the number of recall attempts did not vary significantly (χ^2_7=3.009, p=0.884). Among the 2,120 no-storage participants, there was also no significant difference in using the reminder (χ^2_7=9.727, p=0.205) or successfully recalling the passwords (χ^2_7=9.518, p=0.218).

To understand perceived difficulty, we asked participants whether they agreed with, "Remembering the password I used for this study was difficult." While the above observed metrics for password recall did not vary by condition, perceived difficulty did. Agreement was 47.4% to 49.1% for pattern conditions and 38.5% to 32.6% for the non-pattern conditions. Table IX shows significant differences. Each pattern condition had significantly more difficulty than any nonpattern condition.

6.4. Password Patterns

Next, we examine how participants met and exceeded password-composition requirements.

6.4.1. Common Substrings. Analyses presented in Section 4.5 found eight substrings that were present in at least 1% of Study 1 passwords. This led to the blacklist requirement for some Study 2 conditions. The substring "love" is in 1.4% of passwords created in conditions with the blacklist requirement. No other substring was in 1% or more of these passwords. Containing this substring was correlated with passwords being more likely to be cracked (35.8% to 23.9%) (FET, $p = .006$). This shows that the blacklist requirement did help prevent participants from using common substrings in their passwords. This may help explain why 2s-list12 and 2list12 passwords were less likely to be cracked than 2class12 passwords.

6.4.2. Going Beyond the Requirements. Conditions in Study 2 required either 12 or 16 characters. Of the passwords, 65.4% exceeded their minimum length requirement. These passwords were significantly less likely to be cracked than passwords that did not exceed the minimum (18.9% to 33.8%) (χ^2_1=239.9686, p<.001). Each condition required either two or three different character classes. In addition, 84.3% of passwords exceeded their minimum character-class requirement. These passwords were significantly less likely to be cracked than passwords that did not exceed the minimum (20.1% to 45.4%) (χ^2_1=404.8158, p<.001).

6.4.3. The Pattern Requirement and Character Distribution. The pattern requirement was intended to cause participants to distribute nonlowercase-letter characters throughout their passwords, rather than putting most of them at the beginning or end. To measure how effective this was, we compared character-class distributions in 2pattern12 and 2class12 passwords, which differed only in having the pattern requirement.

The *structure* of a password is a representation of its character classes [Weir et al. 2009]. For example, the structure of password "P4ssword!" is "UDLLLLLLS", where for example "U" indicates an uppercase letter. We examined how often passwords in 2class12 and 2pattern12 had unique patterns. Because 2class12 had more participants, we did not make a direct comparison of the number of unique passwords in each condition. We instead took a random sample of 1,000 participants from each condition and counted how many of them had structures unique among the 1,000. We repeated this experiment 1,000 times. For 2class12, an average of 63.4% of passwords had unique structures. For 2pattern12, it was 81.8%. This is evidence that the pattern requirement led participants to more diverse password structures.

Figure 5 visualizes character-class distributions of 2class12 and 2pattern12. It depicts the character classes of the first and last six characters, because those conditions both require 12 characters. Passwords in 2class12 frequently have special characters at the start and end. While 2pattern12 passwords do not have an even distribution of character classes, they appear to be better mixed than in 2class12. This may explain why the pattern requirement led to stronger passwords.

7. COMPARISON OF CONDITIONS BETWEEN STUDIES

After we conducted both studies and analyzed the results, there were two pairs of conditions that we wanted to compare between the studies. We wanted to compare 2word16 and 2s-list12, because both were more secure than 3class12 without being more difficult to recall. We also wanted to compare comp8 and 2class12, given how similarly 2class12 and 3class12 performed. Both of these comparisons could help us make more specific recommendations to service providers.

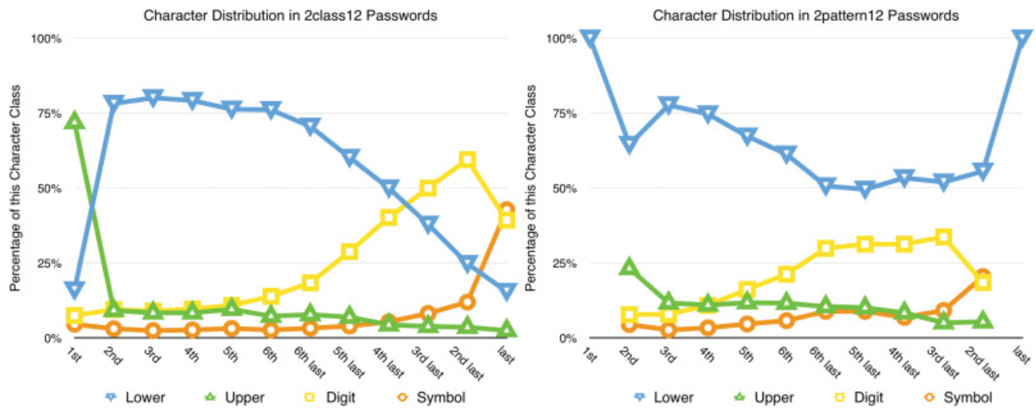

Fig. 5. For the first and last six characters, these graphs show the percentage of each character class in 2class12 and 2pattern12.

We collected data for Study 1 and Study 2 at different times. Therefore, we had concerns about comparing conditions between the studies. To determine whether we could reasonably compare across studies, we compared the two sets of 3class12 participants who finished Part 2 of either study. These two sets of 3class12 participants did not differ significantly in password strength, difficulty with password creation or recall, password storage, or attempts needed to create or recall their passwords ($p \geq .1$). The similarity of 3class12 between studies made us comfortable with making the following two comparisons.

7.1. Comparing 2Word16 and 2S-List12

We compared the 981 2word16 participants who finished Part 2 of Study 1 to the 648 2s-list12 participants who finished Part 2 of Study 2. After 10^6 guesses, their proportions of guessed passwords did not differ significantly (χ_1^2=0.004, p=0.948). After 10^{14} guesses, 2word16 passwords were less likely to be guessed (22.9% to 29.6%) (χ_1^2=8.833, p=0.003). Participants in 2s-list12 were less likely to find password creation difficult (27.3% to 35.2%) (χ_1^2=10.695, p=.001) and took fewer attempts to create their passwords, (1.8 to 2.0 mean attempts) (KW χ_1^2=15.666, p<0). Participants in 2word16 were less likely to store their passwords (51.3% to 56.5%) (χ_1^2=4.045, p=0.044). There was no significant difference in finding password recall difficult (χ_1^2=2.892, p=.089) or Part 2 recall attempts (KW χ_1^2=0.047, p=.829). Neither condition stood out as clearly superior.

7.2. Comparing Comp8 and 2Class12

We compared the 1200 comp8 participants who finished Part 2 of Study 1 to the 661 2class12 participants who finished Part 2 of Study 2. 2class12 passwords were less likely to be cracked than comp8 passwords, after both 10^6 guesses (0.9% to 2.4%) (χ_1^2=4.473, p=0.034) and 10^{14} guesses (35.4% to 49.3%) (χ_1^2=32.566, p<.001). Participants in 2class12 were less likely to find password creation difficult (24.7% to 32.6%) (χ_1^2=12.424, p<.001). They also took fewer password-creation attempts (2.3 to 2.4) (χ_1^2=80.039, p<.001). There was no significant difference in recall attempts for Part 1 (χ_1^2=0.183, p=0.669) or Part 2 (χ_1^2=1.71, p=0.191). There was no significant difference for password-recall difficulty (χ_1^2=2.635, p=0.105). Participants in 2class12 were less likely to store their passwords than those in comp8 (50.8% to

56.9%) (χ^2_1=6.126, p=0.013). This suggests that, similar to 3class12, 2class12 offers both security and usability advantages over comp8.

8. DISCUSSION

We succeeded in finding policies that offer advantages over the traditional comp8 policy, without being worse in any metric we analyzed. We found that forcing uses to break their password-creation habits can increase difficulty, but can also lead to stronger passwords. In this section, we summarize our findings and present recommendations for service providers.

8.1. Results Summary

We conducted two studies on password-composition policies that require longer passwords. Tables I and VI show security and usability metrics from both studies. For each metric, each other condition is compared with the base condition for its study—comp8 for Study 1 and 3class12 for Study 2. We focus on guessnumber thresholds of 10^6 to simulate an online attack and 10^{14} to simulate an offline attack [Florêncio et al. 2014]. The darker red color indicates an unfavorable significant comparison with the base condition, and the lighter blue color indicates a favorable comparison.

Study 1 examined password policies with different minimum lengths, and policies that combined length and character-class requirements. Conditions 3class12 and 2word16 compared favorably to the comp8 policy. Their passwords were easier to create and to recall. Further, their passwords were less likely to be guessed after 10^{14} guesses, and no more likely after 10^6 guesses. Other conditions in the study had advantages, but either compared unfavorably with comp8 in one or more metrics or were not stronger than comp8 after 10^{14} guesses.

Study 1 found evidence that users often met longer-length password requirements in predictable ways. Study 2 examined ways to break users of their predictable password-creation habits for longer passwords. Adding the blacklist or pattern requirement to 2class12 made the resulting passwords more secure. The blacklist requirement made passwords more difficult to create, and the pattern requirement made password creation and recall more difficult and error-prone.

8.2. Creating a Substring Blacklist

A substring blacklist made passwords more secure without making recall more difficult. The ideal contents of a substring blacklist depend on context. For example, passwords in our studies often contained the substring “turk” because we conducted them on MTurk. Most websites would not benefit from having “turk” in their substring blacklist, unless they were websites about Constantinople. Administrators should be sure to include domain-specific keywords, and their service-provider name, in their blacklists.

Given a set of known passwords, creating an optimal set of k substrings to preclude the maximum number of passwords from the set can be reduced to the maximum coverage problem, which is known to be NP-hard. Let each password be an element. Each substring can be considered to be a subset of those passwords. Any password containing that substring is considered to be in the subset of elements associated with that substring. The objective then becomes to create a blacklist of k substrings (or subsets) such that the maximum number of passwords (elements) is covered.

Our technique for creating a substring blacklist from a password corpus might be useful to service providers. We first created a list of all password substrings with lengths between four and the length of the shortest password in the corpus. Then we removed from the list any substring in fewer than 1% of passwords. We made a second pass through the list and removed substrings that did not occur in at least 1% of passwords without being part of a larger substring in the list.

We applied this blacklist-creation algorithm to the 289,039 passwords with at least eight characters in the Yahoo password set [Goodin 2012]. The only two substrings common to at least 1% of these passwords were "1234" and "love." An algorithmic approach to creating a substring blacklist can lead to over-fitting, and we recommend manually improving the blacklist. This can include, for example, adding the name of the website and other terms related to the service. The blacklist that we used in the study would have prevented 12.5% of the passwords in the Yahoo password set.

8.3. The Pattern Requirement

Study 1 indicated that participants required to use special characters usually met those requirements by putting special characters at the start or end of their passwords. This predictable placement can help attackers guess victims' passwords. In Study 2, we tested the pattern requirement, which required passwords to begin and end with lowercase letters. The pattern requirement succeeded in effecting a more even character distribution, as well as an increase in password strength.

However, we also observed that many participants struggled to create and recall passwords that met the pattern requirement. Participants appeared to be so accustomed to putting special characters at the start or end that forcing them to place them within the password caused difficulty. Because of its promising effects on password strength, future work might investigate how this requirement could be made more acceptable to users. For example, it is likely that our participants had never encountered a similar requirement before. Perhaps, after becoming more familiar with it, the requirement would become more usable to them. Further, participants might not have understood the reasoning behind the pattern requirement. It is possible that a better understanding of its function would help users follow the requirement.

8.4. Recommendations for Service Providers

8.4.1. Avoid Using Length-Only Requirements. A number of the password-composition policies we tested required only length. These policies were relatively usable. Many passwords created under these policies were quite strong. However, many participants assigned these policies created very weak, easily guessed passwords. Introducing further requirements to longer-length requirements helped prevent this. Therefore, we recommend against using length-only password requirements, even if a longer length is used.

8.4.2. If You are Using Comp8, Replace It. We chose to study comp8 because it is traditionally considered a "strong" password-composition policy. Our research has shown that there are other policies that are more usable and more secure. We found three policies—2class12, 3class12, and 2word16—that we can directly recommend over comp8. We recommend that an organization using a policy similar to comp8 replace it with one of these.

8.4.3. More Organizations Should be Using Substring Blacklists. Our results are generally positive for including substring blacklists in password-composition requirements. We found that including a blacklist made passwords significantly more secure against offline attacks. While this requirement made some passwords more difficult to create, it did not make their recall more difficult. Many use-cases for passwords involve relatively infrequent password creation, and continual password recall. In these scenarios, a substring blacklist could increase password strength without being a burden on users.

Because substring blacklists make password-creation more difficult, it may not be suited to every service provider. A website or online service wanting to sign up as many users as possible may wish to avoid using a blacklist because it might turn away prospective users. An organization that requires frequent password expiration might also wish to avoid substring blacklists because of the burden they place on users creating passwords. On the other hand, substring blacklists are very well suited to universities having their students and faculty make passwords. It is also well suited to institutions that do not require frequent password expiration. In both cases, a more difficult password-creation process would not turn away users, and it would be infrequent enough not to cause too much frustration. We describe how we created a substring blacklist in Section 8.2.

8.4.4. The Pattern Requirement May be Appropriate for High-Security Service Providers. The pattern requirement requires passwords start and end with lowercase letters. Without this requirement, the vast majority of study participants tended to start or end with required special characters. It appears that users are so accustomed to meeting requirements by putting special characters at the start and end of their passwords that the pattern requirement led to usability difficulty. However, this makes the pattern requirement especially promising for high-security service providers. The pattern requirement increases password strength, and we speculate that it makes users unable to re-use many of their existing "strong" passwords. The pattern requirement, however, does make password create and recall more difficult. Therefore, requiring passwords start and end with lowercase letters may be appropriate for high-security service providers such as banks and email providers.

REFERENCES

Farzaneh Asgharpour, Debin Liu, and L. Jean Camp. 2007. Mental models of computer security risks. In *Proc. WEIS*.

Chris Baraniuk. 2015. Ashley Madison: Two women explain how hack changed their lives. *BBC* Retrieved from http://www.bbc.co.uk/news/technology-34072762.

Bob Beeman. 2004. Using "grep" (a UNIX utility) for Solving Crosswords and Word Puzzle. Retrieved from http://www.bee-man.us/computer/grep/grep.htm#web2.

Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. 2015. Version 1.2 of Argon2. Retrieved from https://password-hashing.net/submissions/specs/Argon-v3.pdf.

Matt Bishop and Daniel V. Klein. 1995. Improving system security via proactive password checking. *Computers & Security* 14, 3 (1995), 233–249.

Joseph Bonneau. 2012. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proc. IEEE Symp. Security & Privacy*.

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

Joseph Bonneau and Ekaterina Shutova. 2012. Linguistic properties of multi-word passphrases. In *Proc. USEC*.

Thorsten Brantz and Alex Franz. 2006. *The Google Web 1T 5-Gram Corpus*. Technical Report. Linguistic Data Consortium.

William E. Burr, Donna F. Dodson, Elaine M. Newton, Ray A. Perlner, W. Timothy Polk, Sarbari Gupta, and Emad A. Nabbus. 2011. *Electronic Authentication Guideline*. Technical Report. NIST.

William E. Burr, Donna F. Dodson, and W. Timothy Polk. 2006. *Electronic Authentication Guideline*. Technical Report. NIST.

Jan Camenisch, Anja Lehmann, and Gregory Neven. 2015. Optimal distributed password verification. In *Proc. CCS*.

Carnegie Mellon University. 2015. Password Guessability Service. Retrieved from https://pgs.ece.cmu.edu.

Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The tangled web of password reuse. In *Proc. NDSS*.

Matteo Dell'Amico and Maurizio Filippone. 2015. Monte Carlo strength evaluation: Fast and reliable password checking. In *Proc. CCS*.

Dave Engberg. 2013. Security Notice: Service-wide Password Reset. Retreived from http://blog.evernote.com/blog/2013/03/02/security-notice-service-wide-password-reset/.

Experian. 2014. Illegal Web Trade of Personal Information Soars to Record Highs. Retrieved from https://www.experianplc.com/media/news/2014/illegal-web-trade-of-personal-information-soars-to-record-highs/.

Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. 2013. On the ecological validity of a password study. In *Proc. SOUPS*.

Dinei Florêncio and Cormac Herley. 2007. A large-scale study of web password habits. In *Proc. WWW*.

Dinei Florêncio and Cormac Herley. 2010. Where do security policies come from? In *Proc. SOUPS*.

Dinei Florêncio, Cormac Herley, and Paul van Oorschot. 2014. An administrator's guide to internet password research. In *Proc. USENIX LISA*.

Dinei Florêncio, Cormac Herley, and Paul Van Oorschot. 2014. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In *Proc. USENIX Security*.

Warwick Ford and Burton S. Kaliski Jr. 2000. Server-assisted generation of a strong secret from a password. In *Proc. WET ICE*.

Dan Goodin. 2012. Hackers expose 453,000 credentials allegedly taken from Yahoo service. Retrieved from http://arstechnica.com/security/2012/07/yahoo-service-hacked/.

Dan Goodin. 2015. Once seen as bulletproof, 11 million+ Ashley Madison passwords already cracked. *Ars Technica*. Retrieved from http://arstechnica.com/security/2015/09/once-seen-as-bulletproof-11-million-ashley-madison-passwords-already-cracked/.

Cormac Herley. 2009. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proc. NSPW*. 133–144.

Cormac Herley and Paul Van Oorschot. 2012. A research agenda acknowledging the persistence of passwords. *IEEE Security and Privacy* 10, 1 (2012), 28–36.

Jun Ho Huh, Seongyeol Oh, Hyoungshick Kim, Konstantin Beznosov, Apurva Mohan, and S Raj Rajagopalan. 2015. Surpass: System-initiated user-replaceable passwords. In *Proc. CCS*. ACM.

Philip Inglesant and M. Angela Sasse. 2010. The true cost of unusable password policies: Password use in the wild. In *Proc. CHI*.

InsidePro. 2005. Dictionaries. Retrieved from http://forum.insidepro.com/viewtopic.php?t=34331. (2005).

Ari Juels and Ronald L. Rivest. 2013. Honeywords: Making password-cracking detectable. In *Proc. CCS*.

Mark Keith, Benjamin Shao, and Paul Steinbart. 2009. A behavioral analysis of passphrase design and effectiveness. *Journal of the Association for Information Systems* 10, 2 (2009), 63–89.

Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Rich Shay, Tim Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Proc. IEEE Symp. Security & Privacy*.

Saranga Komanduri. 2016. *Modeling the Adversary to Evaluate Password Strengh with Limited Samples*. Ph.D. Dissertation. Carnegie Mellon University. CMU-ISR-16-101.

Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. 2011. Of passwords and people: Measuring the effect of password-composition policies. In *Proc. CHI*.

Greg Kumparak. 2013. Vudu Headquarters Robbed, Hard Drives With Private Customer Data Stolen. Retrieved from http://techcrunch.com/2013/04/09/vudu-headquarters-robbed-hard-drives-with-private-customer-data-stolen/.

Bob Lord. 2013. Keeping our users secure. Retrieved from http://blog.twitter.com/2013/02/keeping-our-users-secure.html.

Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. 2014. A study of probabilistic password models. In *Proc. IEEE Symp. Security & Privacy*.

Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. 2013. Measuring password guessability for an entire university. In *Proc. CCS*.

William Melicher, Darya Kurilova, Sean M. Segreti, Pranshu Kalvani, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L. Mazurek. 2016. Usability and security of text passwords on mobile devices. In *Proc. CHI*.

Colin Percival. 2009. Stronger Key Derivation Via Sequential Memory-Hard Functions. http://www.tarsnap.com/scrypt/scrypt.pdf. (2009).

Nicole Perlroth. 2013. LivingSocial Hack Exposes Data for 50 Million Customers. Retrieved from http://bits.blogs.nytimes.com/2013/04/26/living-social-hack-exposes-data-for-50-million-customers/.

John O. Pliam. 2000. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Proc. INDOCRYPT*.

Niels Provos and David Mazieres. 1999. A future-adaptable password scheme. In *Proc. USENIX ATC*.

Ashwini Rao, Birendra Jha, and Gananand Kini. 2013. Effect of grammar on security of long passwords. In *Proc. CODASPY*.

Florian Schaub, Ruben Deyhle, and Michael Weber. 2012. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proc. MUM*.

Bruce Schneier. 2006. MySpace Passwords Aren't So Dumb. Retrieved from http://www.wired.com/politics/security/commentary/securitymatters/2006/12/72300.

SCOWL. 2015. Spell Checker Oriented Word Lists. Retrieved from http://wordlist.sourceforge.net.

Richard Shay, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Alain Forget, Saranga Komanduri, Michelle L. Mazurek, William Melicher, Sean M. Segreti, and Blase Ur. 2015. A spoonful of sugar? The impact of guidance and feedback on password-creation behavior. In *Proc. CHI*.

Richard Shay, Iulia Ion, Robert W. Reeder, and Sunny Consolvo. 2014. "My religious aunt asked why I was trying to sell her Viagra": Experiences with account hijacking. In *Proc. CHI*.

Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Blase Ur, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2012. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proc. SOUPS*.

Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip Seyoung Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2014. Can long passwords be secure and usable? In *Proc. CHI*.

Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2010. Encountering stronger password requirements: User attitudes and behaviors. In *Proc. SOUPS*.

Jens Steube. 2015. Hashcat. Retrieved from https://hashcat.net/oclhashcat/.

Elizabeth Stobert and Robert Biddle. 2014. The password life cycle: User behaviour in managing passwords. In *Proc. SOUPS*.

Elizabeth Stobert and Robert Biddle. 2015. Expert password management. In *Proc. Passwords*.

Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2012. How does your password measure up? The effect of strength meters on password creation. In *Proc. USENIX Security*.

Blase Ur, Saranga Komanduri, Richard Shay, Stephanos Matsumoto, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Michelle L. Mazurek, and Timothy Vidas. 2013. Poster: The art of password creation. In *IEEE Symp. Security & Privacy (Posters)*.

Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2015a. "I Added '!' at the end to make it secure": Observing Password Creation in the Lab. In *Proc. SOUPS*.

Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. 2015b. Measuring real-world accuracies and biases in modeling password guessability. In *Proc. USENIX Security*.

Ashlee Vance. 2010. If Your Password Is 123456, Just Make It HackMe. The New York Times, http://www.nytimes.com/2010/01/21/technology/21password.html. (January 21, 2010).

Rafael Veras, Christopher Collins, and Julie Thorpe. 2014. On the semantic patterns of passwords and their security impact. In *Proc. NDSS*.

Rafael Veras, Julie Thorpe, and Christopher Collins. 2012. Visualizing semantics in passwords: The role of dates. In *Proc. VizSec*.

Emanuel von Zezschwitz, Alexander De Luca, and Heinrich Hussmann. 2014. Honey, I shrunk the keys: Influences of mobile devices on password composition and authentication performance. In *Proc. NordiCHI*.

Rick Wash. 2010. Folk models of home computer security. In *Proc. SOUPS*.

Charles Matthew Weir. 2010. *Using Probabilistic Techniques To Aid In Password Cracking Attacks*. Ph.D. Dissertation.

Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proc. CCS*.

Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password cracking using probabilistic context-free grammars. In *Proc. IEEE Symp. Security & Privacy*.

Yulong Yang, Janne Lindqvist, and Antti Oulasvirta. 2014. Text entry method affects password security. In *Proc. LASER*.

Anjie Zheng. 2015. VTech Has Yet to Put a Price on Hack, Chairman Says. *Wall Street Journal*. Retrieved from http://www.wsj.com/articles/vtech-has-yet-to-put-a-price-on-hack-chairman-says-1449556689. (December 8, 2015).

Received May 2015; revised December 2015; accepted February 2016