Practical Recommendations for Stronger, More Usable Passwords Combining Minimum-strength, Minimum-length, and Blocklist Requirements

Joshua Tan, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor
Carnegie Mellon University
{jstan,lbauer,nicolasc,lorrie}@cmu.edu

ABSTRACT

Multiple mechanisms exist to encourage users to create stronger passwords, including minimum-length and character-class requirements, prohibiting blocklisted passwords, and giving feedback on the strength of candidate passwords. Despite much research, there is little definitive, scientific guidance on how these mechanisms should be combined and configured to best effect. Through two online experiments, we evaluated combinations of minimum-length and character-class requirements, blocklists, and a *minimum-strength* requirement that requires passwords to exceed a strength threshold according to neural-network-driven password-strength estimates.

Our results lead to concrete recommendations for policy configurations that produce a good balance of security and usability. In particular, for high-value user accounts we recommend policies that combine minimum-strength and minimum-length requirements. While we offer recommendations for organizations required to use blocklists, using blocklists does not provide further gains. Interestingly, we also find that against expert attackers, character-class requirements, traditionally associated with producing stronger passwords, in practice may provide very little improvement and may even reduce effective security.

CCS CONCEPTS

• **Security and privacy** → **Authentication**; **Usability in security and privacy**.

KEYWORDS

password policies; neural networks; blocklists

ACM Reference Format:
Joshua Tan, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2020. Practical Recommendations for Stronger, More Usable Passwords Combining Minimum-strength, Minimum-length, and Blocklist Requirements. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3372297.3417882

1 INTRODUCTION

To help users create stronger passwords, system administrators often require passwords to exceed a certain length, contain at least a specific number of character classes, or not appear on a blocklist [19]. Users are also often nudged to create stronger passwords by password meters that give feedback on the strength of candidate passwords and suggestions about how to improve them.

Early guidance for how to deploy these approaches relied mostly on common sense and experts' opinions [17, 18]. Over the past decade, a scientific basis has emerged for what requirements are most effective at encouraging users to create passwords that are strong but still memorable. For example, research has shown that increasing minimum length may increase password strength more than relying just on character class requirements [26]; that password meters can very effectively nudge users to create stronger passwords [28]; and that carefully configured blocklists can help prevent users from picking easily guessed passwords [8].

These early efforts shed light on which password requirements were more or less effective, but stopped short of providing empirically evaluated, definitive guidance for how to combine requirements. In this paper, we seek to address this. Building on previous findings, we empirically examine combinations of length, character-class, blocklist, and password meter requirements—all of which were previously individually studied—as well as minimum-strength requirements, which have been less studied. We consider practical implementions of minimum-strength requirements, using estimates from a neural network trained on leaked password data. Our research questions examine and compare the security and usability properties of differently configured blocklists and minimum-strength requirements, with careful attention to how the choice of underlying composition requirements affect these properties.

Our results, derived through two successive experiments that investigated a wide range of potential interactions between requirements, allow us to provide concrete, practical recommendations for how to combine and configure these mechanisms. We find that how users pick passwords has changed over time, and that this, in combination with advances in password guessing, implies that requiring passwords to have multiple character classes brings at best minor benefit to password strength. Although some blocklist configurations are more effective than others at eliminating weak passwords, policies that require passwords to have at least eight characters and that simultaneously prohibit passwords that can be guessed within 10^6 guesses perform better—in terms of encouraging password strength—than the best-performing blocklist policies we examine. Properly configured minimum-strength policies not only match our top tested blocklist policies in usability, but can also provide better security, especially against offline attacks that make up to 10^{14} guesses. For organizations that nevertheless require blocklists, we recommend blocklist policies that impose a minimum of eight characters and perform a fuzzy blocklist-matching check against either a subset of the most popular passwords found in password leaks, or that perform a fullstring-matching check against a very large database of publicly-leaked passwords.

A primary contribution of our work is to design, evaluate, and recommend configurations for minimum-strength requirements. These requirements build on the large body of password research conducted over the past 15 years. Conceptually, minimum-strength requirements represent the goal of other types of password-creation requirements that have been proposed and tested: to reject weak, easily-guessed passwords while permitting strong, hard-to-guess passwords. In this study, we present a concrete (neural-network-driven) implementation of minimum-strength requirements that both faithfully achieves password-strength goals and achieves comparable usability to the best competing blocklist-based policies we tested. Neural-network-driven minimum-strength requirements are easy to deploy, flexibly configurable, and can be easily retrained to reflect changing patterns in passwords over time.

2 BACKGROUND AND RELATED WORK

Here we discuss related work on password composition policies and blocklists, and on measuring password strength, including to implement data-driven minimum-strength requirements.

2.1 Composition policies

Password composition policies aim to help users create more unique, less predictable passwords. Composition policies can be used at password-creation time to enforce a minimum number of character classes—uppercase letters, lowercase letters, symbols, and digits—and a minimum-length requirement.

Early work on composition policies focused on character-class requirements to increase password strength [18]. In general, policies that require more character classes have been found to produce overall stronger passwords [11, 13]. Later work explored policies that emphasized length over character-class composition. Researchers exploring length requirements have found that reducing the number of required character classes while increasing the minimum-length requirement could strengthen passwords without decreasing their memorability or making them more difficult to create [11, 13, 25, 26].

Although researchers and, more recently, NIST advise to avoid composition policies requiring a minimum number of character classes, this type of policy is still often used in practice [4, 14]. We include such policies in our studies to provide concrete recommendations to organizations that may continue to rely on them.

2.2 Blocklists

Even if password requirements are generally effective at improving strength, some users will fulfill them predictably. Thus, character-class and minimum-length rules are insufficient to prevent very weak passwords [25]. For example, *4class8* and *1class16* policies allow extremely predictable passwords such as "Password1!" and "passwordpassword." A common mitigation is to combine composition requirements with a blocklist check. For instance, NIST 800-63B recommends that passwords not be in a list of commonly-used, expected, or compromised values [19]. Properly configured blocklist checks can reject predictable, or easy-to-guess passwords.

A blocklist requirement uses a wordlist and a matching algorithm that checks whether a given password is prevented by that wordlist. Wordlists can contain common sequences of characters, as well as previously-leaked passwords. Matching algorithms range from exact match (a candidate password should not be in the wordlist), to more complicated rules—e.g., stripping symbols and digits from the password, making it case-insensitive, and checking that the resulting string does not match any wordlist entry.

Prior work found that blocklists used by major online service providers vary in architecture, including client-side, server-side, and hybrid approaches [4]. Blocklists differed in their wordlists and matching algorithms, e.g., whether and how symbols or digits were removed from passwords before matching. They also differed in whether they forbid, warned, or decreased a password-strength score if a blocklist check returned positive.

Initial password research focused on blocklists that forbid dictionary words in passwords. For example, many studies have examined policies that perform blocklist checks against the OpenWall wordlist [12]. These studies found that blocklist checks caused more annoyance compared to policies without them. More recently, blocklists based on ad-hoc cracking dictionaries were found to produce weaker passwords than when relying on larger wordlists gleaned from password leaks, such as the Xato password corpus [5]. Blocklists based on Xato were found to produce reasonably strong passwords [8, 28]; we will use this wordlist here.

By analyzing large leaked sets of passwords, Weir et al. found that larger blocklists strengthened passwords [33]. Kelley et al. tested three blocklists varying in size and similarly found that larger blocklists produced stronger passwords [11]. While blocklists might be effective against online attacks, Florêncio et al. questioned their practicality against offline attacks due to the required wordlist size or potential negative usability impact [7].

Despite these studies, our understanding of blocklist usability and security effects is incomplete, particularly for large blocklists or string-matching algorithms that remove non-alphanumeric characters. Prior work that uses retrospective policy analyses has limitations, e.g., subsetting policy-allowed passwords retroactively overestimates the impact of password policies on strength [14] and cannot account for users who replace blocklisted passwords with even weaker, but non-blocklisted ones. Retrospective studies also cannot analyze many usability-related aspects of password policies.

Prior findings may also not apply to modern contexts. Conclusions on blocklist usability derived from studies that lack real-time feedback [11, 13] may lead to overly pessimistic usability results. For example, the largest cause of policy compliance failure in prior work was the dictionary check, which real-time feedback could alleviate [25]. Additionally, blocklists may differently impact password strength and usability depending on interactions between the wordlist, matching algorithm, and composition policy they augment. For instance, prior work has examined digit-and-symbol-stripping matching algorithms with *3class8* and *4class8* policies, and wordlists based on the OpenWall dictionary, which was found to be relatively ineffective [12, 13].

2.3 Quantifying password strength

The strength of passwords produced under a particular password-creation policy can be quantified using guess numbers. Guess numbers can be computed by enumerating the passwords predicted by a particular guessing model, in decreasing probability, or according to the order that a commonly-configured password-cracking tool would output guesses. A password's guess number estimates the number of guesses an attacker need make before guessing that particular password. Guess numbers are parameterized by the particular tool used, as different tools and models guess passwords in different orders. Prior work recommends taking the minimum guess number across multiple automated guessing methods as a conservative proxy for an expert attacker's guessing capabilities [30].

The Password Guessability Service (PGS) [21] is a state-of-the-art tool for estimating password strength. PGS supports the popular password-cracking tools Hashcat and John the Ripper, as well as tools based on password-modeling approaches such as Probabilistic Context-Free Grammars (PCFG) and artificial neural networks. PGS also provides *min-auto* guess numbers, computed as the lowest guess number among the set of supported password-guessing methods. For probabilistic models, Monte Carlo sampling methods allow for estimation of guess numbers for low-probability passwords [5]. Probability-to-guess-number mappings can be precomputed, enabling client-side, real-time guess number estimates. We apply this method in our study, using a meter based on prior work [28].

Guess numbers do not paint a complete picture. The number and frequency of guesses an attacker can make depend on many factors, including whether it is an online or offline attack, the extent to which rate limiting is applied, and whether defenses such as iterated hashing are deployed. Florêncio et al. discuss the "online-offline chasm" between password-strength thresholds that may be relevant in practice [7]. They argue that offline attacks may not always be a threat, e.g., for service providers that reversibly encrypt passwords or store passwords in plaintext. When offline attacks are applicable, they argue that the user effort to create passwords that resist such attacks is usually wasted unless passwords can withstand attacks that make up to 10^{14} guesses.

2.4 Minimum-strength requirements

Minimum-strength password requirements have been explored less than other types of policy requirements. Accurate strength estimates for individual passwords are indeed difficult to perform in real-time. Password-strength heuristics can roughly estimate password strength, but the accuracy of these estimates may be insufficient. Prior work evaluating password meters has found that strength estimates from heuristic-based meters are often inconsistent [4] and contradict guess number estimations [16]. One of the more accurate password-strength estimators, zxcvbn [34], uses advanced heuristics to output quite reliable password strength estimates at low guess numbers typical of online attacks. Although zxcvbn can be configured to meet minimum password-strength thresholds, this has not been evaluated through user studies.

Recently, Melicher et al. designed a client-side recurrent neural network for modeling password strength [16]. This development has enabled minimum-strength requirements based on password-strength estimates that are both accurate and data driven, rather than heuristic driven. In our study, we explore minimum-strength requirements expressed as a minimum guess number estimated by a neural network. This is the first time we are aware of that this type of requirement has been incorporated into a password-creation policy and evaluated in a user study.

3 METHODOLOGY

Here we present our experimental factors and conditions; the design of the user studies we used to collect data; and the statistical methods we applied to analyze that data. To limit the number of interactions between policies that we would have to simultaneously explore, we conducted two experiments (*Experiment 1* and *Experiment 2*), each involving independent data collection. Both were identical in terms of methodology and implementation—only the experimental conditions differed.

3.1 Experimental factors

Each of our user studies presented participants with password-creation policies that differed based on assigned treatment. The experimental factors consisted of three types of requirements that can be enforced by a password-creation policy: composition, blocklist, and minimum-strength requirements.

We performed retrospective analyses on randomly-selected subsets of leaked 000webhost passwords to help identify the parameters to explore in our user study for each type of experimental factor. This involved retroactively applying a password policy to a set of leaked passwords and observing the proportions and strengths of passwords that were allowed or rejected by that policy. Retrospective analyses and analyses relying on leaked data have inherent limitations discussed in Section 2.2. The overall findings we present are based on data collected from *experimentally* designed user studies, which avoids these limitations. In addition, results from Experiment 1 informed the parameters explored in Experiment 2.

Composition requirements. All policies required passwords to be at least eight characters long. In addition, some policies required longer password lengths or required passwords to contain a minimum number of character classes. We abbreviate composition requirements using a notation that lists the required number of character classes followed by the minimum length, e.g., *3c12* corresponds to requirements that passwords contain at least three character classes and at least twelve characters. In Experiment 1, we tested *1c8*, *1c16*, *3c8*, *3c12*, and *4c8*, each of which has been explored in prior work [11, 13, 15, 18, 26, 27]. In Experiment 2, we explored additional longer-length one-class policies (*1c10* and *1c12*).

Blocklist requirements. We tested policies incorporating a blocklist requirement, which rejected any password that matched an entry on a list of prohibited strings. We explored several wordlists and matching algorithms. The majority of our blocklist configurations followed previous work [8, 28, 34] in using the *Xato* wordlist, consisting of 96,480 passwords that appeared four or more times in the leaked Xato corpus of 10 million passwords [3]. We also used a wordlist of 555 million unique passwords previously leaked in data breaches that are accessed using the freely available *Pwned* Passwords API [10].[1] Last, we tested the wordlist (and matching algorithm) used at Carnegie Mellon University (CMU), which consisted of 630,034 English dictionary words [1, 20].

We tested four matching algorithms: case-insensitive full-string (*cifs*); case-sensitive full-string (*fs*); stripping digits and symbols and then performing a case-insensitive full-string comparison (*strip-cifs*); and checking whether any length-5 substring of any wordlist entry was a case-insensitive substring of the candidate password (*ciss*). Each of these has been used in deployed password-creation policies [4]. The *ciss* algorithm has been explored by prior work [25], albeit with a much smaller wordlist than we considered.[2]

Minimum-strength requirements. In addition to composition and blocklist requirements, we tested policies incorporating a minimum-strength requirement, expressed as the minimum number of guesses that passwords should withstand in a guessing attack. We used password-strength estimates computed by a client-side, JavaScript-based neural network, implemented and trained following the approach of Melicher et al. [16] (see Appendix A). We defined minimum-strength requirements in terms of a log-10 guess number threshold. For example, *NN6* required passwords to have password-strength estimates no weaker than 10^6 guesses.

We tested four minimum-strength thresholds, ranging from 10^6 to 10^{12} guesses. In Experiment 1 we tested policies that included *NN6* and *NN12* requirements. Results suggested that *NN6* requirements may be too lenient for protecting high-value accounts and that *NN12* requirements can make password-creation annoying or difficult for some users. Hence, we tested *NN8* and *NN10* in Experiment 2 to explore minimum-strength thresholds that might produce a better balance of both security and usability.

3.2 Research questions

We designed our experimental conditions to answer specific research questions. Some research questions explored the security and usability impacts of policies differing in a single experimental factor (e.g., blocklist configuration). We also investigated whether certain policy components impact password strength or usability differently depending on the configuration of other policy components (e.g., the same blocklist configuration used alongside different composition policies). Table 1 lists our research questions and the experimental conditions and comparisons used to answer them.

Baseline	Comparisons	Exp.
*RQ1: What is the impact of **adding a blocklist** to 1c8 and 3c8?*		
1c8	1c8+Pwned-fs, 1c8+Xato-cifs, 1c8+Xato-strip-cifs, 1c8+Xato-ciss	1
3c8	3c8+Xato-cifs	1
*RQ2: What is the impact of varying **blocklist reqs.** on 1c8?*		
1c8+Xato-strip-cifs	1c8+Pwned-fs, 1c8+Xato-cifs, 1c8+Xato-ciss	1
1c8+Xato-strip-cifs	1c8+Pwned-fs	2
*RQ3: What is the impact of varying **char-class and min-length reqs.** on NN6?*		
3c8+NN6	1c8+NN6, 1c16+NN6, 2c12+NN6, 3c12+NN6	1
*RQ4: How do **min-strength reqs. compare with blocklists**?*		
1c8+NN6	1c8+Pwned-fs, 1c8+Xato-cifs, 1c8+Xato-strip-cifs, 1c8+Xato-ciss	1
1c8+Pwned-fs	1c8+NN8, 1c8+NN10	2
1c8+Xato-strip-cifs	1c8+NN8, 1c8+NN10	2
*RQ5: How do **min-strength reqs. interact with min-length reqs.**?*		
1c8+NN10	1c8+NN8, 1c10+NN10	2
1c10+NN8	1c8+NN8, 1c10+NN10	2
1c12+NN10	1c8+NN10, 1c10+NN10	2
*RQ6: How do **blocklist reqs. interact with char-class reqs.**?*		
1c8+Xato-strip-cifs	4c8+Xato-strip-cifs	2
1c8+Pwned-fs	4c8+Pwned-fs	2
4c8+Xato-strip-cifs	4c8+Pwned-fs	2

Table 1: Research questions and planned comparisons.

3.2.1 *Experiment 1.*

We tested 15 experimental conditions designed to answer four high-level research questions. In order to both quantify the impact of blocklists relative to policies that only required composition requirements (*RQ1*) and to find blocklist requirements for use in *1c8* policies that performed well on both security and usability dimensions (*RQ2*) we tested blocklist configurations that were either commonly used or recommended by prior work. Our third high-level research question focused on the impact of character-class and minimum-length requirements on *NN6* minimum-strength policies (*RQ3*). The particular set of character-class and minimum-length combinations we explored included composition policies explored in prior work. Our fourth research question involved directly comparing blocklist and minimum-strength policies on both security and usability dimensions (*RQ4*). In Experiment 1, we explored this question by comparing a variety of blocklists with a *NN6* minimum-strength policy that we hypothesized would provide adequate protection against online attacks, withstanding at least 10^6 guesses (*RQ4.A*).

3.2.2 *Experiment 2.*

The results of Experiment 1 raised additional research questions that could not be answered with the experimental data that had already been collected. Therefore, we conducted a second experiment, testing seven new conditions and re-testing three conditions from Experiment 1.[3] One goal of Experiment 2 was to explore how specific minimum-length requirements interacted with specific minimum-strength requirements to effect usability or security (*RQ5*). In particular, we hypothesized that longer minimum-length requirements could make minimum-strength requirements easier to satisfy. We explored this question using strength thresholds in-between those we tested in Experiment 1; results from that experiment had suggested *NN8* and *NN10* requirements may

[1]This API employs privacy-protecting mechanisms to protect the confidentiality of submitted passwords: it only accepts SHA-1 hashes of passwords and utilizes a k-anonymity range search to report matches [9].

[2]We use a computationally efficient *ciss* implementation that performs multiple substring searching via the Rabin-Karp algorithm [2].

[3]We collected new data for each policy in Experiment 2, even if that policy had been previously tested in Experiment 1.

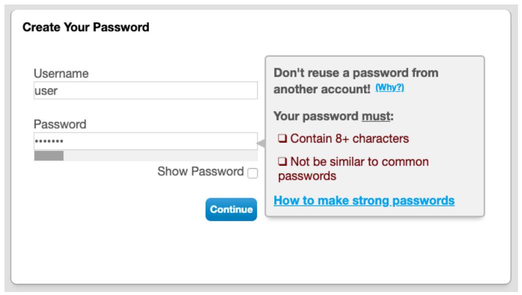
provide the level of offline protection needed for high-value accounts. Experiment 2 was also designed to test whether blocklist requirements impact password strength or policy usability differently depending on the particular character-class requirements they are combined with, and vice versa (*RQ6*). We hypothesized that fullstring blocklist checks against lists of leaked passwords might be less useful for policies requiring many character classes, since leaked passwords may be less likely to contain many character classes. We also hypothesized that the *strip-cifs* matching algorithm might be especially frustrating to users when combined with a *4c8* policy; compared to *1c8* passwords, 4c8 candidate passwords might be more likely to incorporate digits and symbols in ways that would be rejected by blocklist checks that first strip digits and symbols. Lastly, we revisited *RQ4* comparing top-performing blocklist policies from Experiment 1 with the additional *NN10* and *NN12* policies tested in Experiment 2 (*RQ4.B*).

3.3 User-study protocol

For each experiment we ran a user study on Amazon Mechanical Turk in which participants were tasked with creating and recalling a password under a randomly assigned password policy. The design of our user studies closely followed that of prior work [13, 26, 28]. In Part 1, participants were asked to role play, imagining that they needed to create a new password because their main email account provider had been breached. We emailed participants two days later asking them to participate in Part 2, in which they were asked to recall their password. We considered the data of only the participants who completed Part 2 between two and five days after Part 1. After each part, participants completed a survey that collected demographic and usability-related data. The survey materials are provided in Appendix E.

The password-creation task in Part 1 used a password meter developed in prior work, which incorporated real-time requirements feedback, a password-strength bar, and text feedback on improving password strength. Participants were shown feedback on improving password strength only after all composition, minimum-strength, and blocklist requirements were satisfied. The password meter's configuration was based on best practices empirically shown by prior work [28]. We communicated unmet minimum-strength and blocklist requirements as follows: for the *Xato* blocklist configurations the meter reported that the password must "not be an extremely common password;" for the *Pwned* blocklist configurations that the password must "not use a password found in previous security leaks;" and for the minimum-strength requirements that the password must "not be similar to common passwords" (Figure 1).

We submitted the passwords created by participants to PGS [21], which computed guess numbers for each PGS-supported guessing approach using its recommended configuration. We additionally computed guess numbers using a set of neural networks (collectively referred to as the PGS3 NN) that we trained ourselves, closely following the design and implementation of password models in prior work [16]. When computing min-auto guess numbers, we selected each password's lowest guess number among all guessing approaches. For the NN guessing approach, we use PGS3 NN guess numbers in place of PGS-reported NN guess numbers, given the improved guessing performance of the PGS3 NN (see Appendix A).


Create Your Password

Username
user

Password
•••••••

Show Password

Continue

Don't reuse a password from another account! (Why?)

Your password must:

❑ Contain 8+ characters

❑ Not be similar to common passwords

How to make strong passwords

Figure 1: Password-creation meter displaying unmet password policy requirements.

In addition to evaluating the strength and objective usability (e.g., memorability) of passwords created under each policy, we wanted to understand their usability in terms of user difficulty or frustration when creating or recalling passwords. Participants' responses to surveys shown after both Part 1 and Part 2 shed light on this. Our surveys also asked questions such as whether participants reused a previous password or wrote their password down after creating it. In order to elicit truthful responses we told participants that they would receive compensation regardless of their answers.

We instrumented our study to record password-creation and recall keystrokes and whether participants copied and pasted their password during recall tasks. When analyzing password recall, we only analyze data for participants who: typed in their password from memory (as self-reported in the survey); said they didn't reuse their study password (as self-reported); and didn't copy and paste their password during the Part 2 recall task, either manually from a file or using a password manager/browser (based on keystroke data). Study participants who become frustrated with password-creation requirements may be more likely to drop out of our study. We record and analyze dropout rates between experimental conditions as potential evidence of usability issues for a given policy.

The full set of usability metrics we considered include both objective (creation/recall time, recall success, study dropout, copy/paste from storage/password managers) and subjective data (creation annoyance/difficulty, difficulty remembering).[4] Each of these metrics have been used in prior work to measure usability impacts of password-creation policies [24, 27, 28].

We recruited study participants from Mechanical Turk (MTurk). Workers were required to be located in the United States, have had at least 500 HITs approved, and have a HIT approval rate of 95% or higher. Workers were not allowed to participate in our study more than once. We paid 55 cents for Part 1 of our study and 70 cents for Part 2. Our study protocol was approved by our institutional review board and all participants completed online consent forms.

Experiment 1 participants were recruited in July and August 2019. Their ages ranged from 18 to 81 years, with a median of 35. 53% of participants were female and 47% male. Of the 5,099 participants who started the study, 4,317 finished Part 1 and 3,463 also finished Part 2. Most (81%) participants reported that they did not have a

[4]As we employ real-time feedback in our password meter, our study data do not include the notion of a password-creation attempt. However, this concept is closely related to creation time and creation annoyance/difficulty.

technical degree or work in an area related to computer science or information technology. Experiment 2 participants were recruited in October and November 2019. Their ages ranged from 18 to 90 years, with a median of 35. 56% of participants were female, 43% male, 1% reported their gender as "Other," and the remainder chose not to answer. Of the 4,817 participants who started the study, 4,005 finished Part 1 and 3,014 also finished Part 2. Our password-recall analysis includes data for 1,518 participants in Experiment 1 and 1,362 participants in Experiment 2, excluding those who reported reusing a password or not entering their password from memory.

3.4 Statistical analysis

Before running each experiment, we identified a set of hypothesis tests we planned to conduct to answer our research questions. We perform omnibus tests to compare three or more conditions as well as pairwise tests.[5] For each family of tests (the combination of test type and research question), we chose the baseline condition to be used in pairwise comparisons before collecting data.

To compare the overall strength of passwords created under different policies, we use an extension of the Log-rank test called the Peto-Peto test (PP). This test, used in prior work [14], weighs early-appearing differences in guess curves more heavily than later differences, corresponding to heavier weight for strength differences that resource-constrained or rate-limited attackers could exploit. The Peto-Peto test is also appropriate when many data points are censored. In our study, passwords with guess numbers past our offline attack threshold of 10^{14} were censored prior to the test (i.e., labeled as unguessed), as we wanted to compare password guessability only up to the number of guesses that a typical attacker could feasibly attempt in an offline attack.

To compare the vulnerability of passwords to guessing attacks of different magnitudes, we apply Chi-square tests of independence and Fisher's exact tests (FET) to the percent of passwords in each set that an attacker would guess within 10^6 and within 10^{14} attempts.[6] These thresholds have been used in prior work as estimates of how many guesses an online and an offline attacker could make [7], respectively. Unless otherwise noted, analyses that operate on guess numbers are based on min-auto guess number estimates.

We examine usability through statistical tests of Part 1 and Part 2 survey data (password-creation sentiment, post-creation actions) and behavioral data collected by our study framework (study dropout rates, password-creation time, Part 2 recall time, and Part 2 recall success). We bin categorical and Likert data before applying Chi-square tests and Fisher's tests (e.g., Likert agreement data is grouped into two bins: "Strongly agree" or "Agree" vs. otherwise). For comparing count data, we use the non-parametric Kruskal-Wallis (KW) and Mann-Whitney U (MWU) tests.

We record whether text entered into the password-creation field failed to meet requirements, but the real-time nature of requirements feedback in our meter means that even if a blocklist or minimum-strength requirement was unsatisfied at some point, the participant may not have intended to actually create that password—they may have had a different password in mind and hadn't finished typing it. To shed light on whether participants actually encountered one of these unmet requirements for a password, our survey asked "were any passwords you tried to create rejected for the specific reason shown above?" We interpret affirmative answers as evidence that those participants changed their password at least once due to the associated policy requirement.

Within each family of tests, we only perform pairwise tests if the corresponding omnibus test is statistically significant. We use the Holm-Bonferroni method to correct for multiple pairwise comparisons within each family and report adjusted p-values. All hypothesis tests use a significance level of 0.05. When comparing two policies, we only attribute differences to a particular policy dimension if all other dimensions in those policies are identical.

3.5 Limitations

Our study has limitations common to user studies conducted on MTurk. Study participants may not have created passwords similar to those they would have created for actual high-value accounts, despite our role-playing instructions. However, prior work has shown that MTurk passwords collected in this way are similar to actual user passwords created for high-value accounts [6, 14].

Our password-policy results and recommendations rely on passwords being created under the specific password meter we used in our study. This meter provided text feedback on how to improve passwords, a strength bar, and real-time requirements feedback, each of which was configured according to recommendations from prior studies [25, 28, 29]. Based on survey responses, the majority of participants found the meter to be informative, helpful, and influential. For example, most participants reported that they implemented changes suggested by text feedback and that it was important to them that the colored bar rated their password highly. Experiments using password meters with substantially different implementations may produce different results.

It is worth noting that our analysis and recommendations concerning blocklists do not apply to site-specific or user-specific blocklists, which are useful for preventing passwords based on user-associated data or contextual information that targeted guessing attacks could leverage (e.g., user IDs, words related to the service).

4 RESULTS

The results we report here lead to our recommendation for password policies that include both minimum-length and minimum-strength requirements. In case an organization decides against minimum-strength requirements, we recommend two policies incorporating minimum-length and blocklist requirements. These policies provide less protection than minimum-strength policies against offline attacks, but provide adequate protection against online attacks while remaining usable during password creation.

Our results from Experiment 1 show that blocklists may not improve password strength substantially if the blocklist check uses a strict matching algorithm with an insufficiently large wordlist. However, when properly configured, either blocklist requirements

[5] We tried a Cox regression model to measure guessability differences but opted for pairwise hypothesis tests instead, due to poor fit of the linear model to our data.

[6] Our conservative assumption is that the attacker knows the password distribution and makes guesses in order of decreasing probability. While we assume the attacker knows the length and character-class requirements when making guesses, we do not assume that the attacker knows which passwords would have been rejected by blocklist or minimum-strength requirements in order to avoid guessing those passwords.

or minimum-strength requirements can be combined with other requirements to provide adequate protection against online guessing attacks. In Experiment 2 we explore in more depth *1c8* minimum-strength policies that provide strong protection against both online and offline guessing attacks. We also extensively analyze interaction effects between policy components. Experiment 2 results show that *NN8* and *NN10* policies can be just as usable as the blocklist policies we test, while also producing passwords more resistant to offline attacks. In this section, we describe the results from both experiments, organized by research question. P-values for each statistical test can be found in Appendix F.1.

4.1 RQ1: Impact of blocklists

We compared each blocklist condition to its corresponding *1c8* or *3c8* baseline condition to quantify the impact of blocklists on guessability and usability. **We found blocklist configurations *1c8+Pwned-fs* and *1c8+Xato-strip-cifs* significantly improved password strength over their baseline without substantial harm to usability.**

As shown in Figure 2, passwords created under either *1c8+Xato-cifs* or *3c8+Xato-cifs* were neither stronger overall nor less likely to be guessed in online attacks than passwords created under the baseline policies that only contained composition requirements. While blocklist policies that use full-string matching can provide adequate protection against online guessing attacks (as demonstrated by *1c8+Pwned-fs*), our results suggest that this requires a much larger wordlist than the *Xato* wordlist we tested.

Of the policies with blocklists that improved password defense against online attacks, two policies did so without also making passwords substantially more difficult or time-consuming to create. Both *1c8+Pwned-fs* and *1c8+Xato-strip-cifs* passwords were much less likely to be guessed in online attacks (within 10^6 guesses) than *1c8* passwords (FET: 0% and 1% guessed, resp., vs. 6% guessed). Yet, participants did not find either policy substantially more annoying or difficult relative to a *1c8* policy (see Figure 10 in Appendix).

4.2 RQ2: Blocklist reqs. for *1c8* policies

We next compared *1c8* blocklist policies. All pairwise comparisons were made with respect to *Xato-strip-cifs*, which was recommended in prior work [8]. We found that **case-sensitive, full-string matching against very large blocklists of leaked passwords leads to similarly usable and secure passwords as fuzzy matching against smaller blocklists of the most common leaked passwords.**

Prior work hypothesized that the *strip-cifs* algorithm would produce strong passwords by preventing simple modifications to blocklisted passwords that might pass the blocklist check without improving password strength [8]. Our user study confirms this. As shown in Figure 2, passwords created under the *1c8+Xato-cifs* policy, which did not strip digits and symbols before performing blocklist checks, were overall weaker and more susceptible to online guessing than passwords created under the *1c8+Xato-strip-cifs* policy, which stripped digits and symbols (PP, FET: 5% vs. 1% guessed). Furthermore, while the stricter matching algorithm used in *1c8+Xato-strip-cifs* led to slightly longer password-creation times

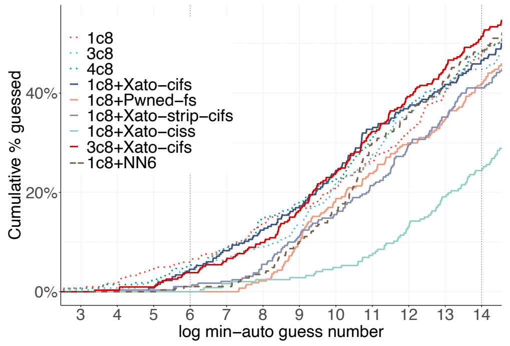


Figure 2: Min-auto guess numbers for Experiment 1 blocklist, composition-requirements-only, and *1c8+NN6* policies.

compared to *1c8+Xato-cifs* (MWU: median of 93 s vs. 70 s), it did not make password creation more challenging or annoying.

Among blocklist configurations using the same *Xato* wordlist, only *1c8+Xato-ciss* produced overall stronger passwords that were more resistant to 10^{14} offline attacks than *1c8+Xato-strip-cifs* (PP, FET: 24% vs. 41% guessed). However, as shown in Figure 10 and Table 2, severe password-creation usability issues associated with *1c8+Xato-ciss* prevent us from recommending it in place of *1c8+Xato-strip-cifs*. Participants took longer to create passwords under *1c8+Xato-ciss* than under *1c8+Xato-strip-cifs* (MWU: median of 139 s vs. 93 s) and reported more annoyance (FET: 47% vs. 35%) and difficulty (FET: 49% vs. 27%). Compared to *1c8+Xato-strip-cifs* participants, *1c8+Xato-ciss* participants were also more likely to drop out before finishing Part 1 (FET: 26% vs. 12%) and to digitally store or write down their password after creating it (FET: 65% vs. 53%). These results lead us to conclude that while a *ciss* blocklist matching algorithm can provide strong security against guessing attacks, it also may severely harm password-creation usability if used alongside a wordlist as large as or larger than the *Xato* wordlist.

Besides *Xato*-based blocklists, we tested a blocklist configuration that used *fs* matching with the much larger *Pwned* wordlist. We found that *1c8+Pwned-fs* and *1c8+Xato-strip-cifs* led to passwords of similar strength, both in terms of overall password strength and in terms of resistance to online and offline guessing attacks. Participants also reported similar usability during password creation. Although *1c8+Xato-strip-cifs* participants were much more likely to report noticing a password they wanted to create being rejected by the blocklist requirement than *1c8+Pwned-fs* participants (FET: 50% vs. 23% noticed), they did not take substantially longer to create their password nor report more difficulty or annoyance. Thus, we conclude that blocklists that perform *fs* checking against the *Pwned* wordlist can provide comparable protections against guessing attacks and similar usability compared to blocklists that perform *strip-cifs* checking against the *Xato* wordlist.

4.3 RQ3: Composition requirements for min-strength policies

We examined whether certain combinations of minimum-strength and composition requirements would lead to stronger or easier-to-create passwords. As we had hypothesized that a *NN6* requirement

Condition	# in Part 1	# in Part 2	Part 1 dropout	Creation time	Creation difficult	Creation annoying	Guessed @ 10^6	Guessed @ 10^{14}	Noticed reject	Stored pwd	Recall success	Recall time
Experiment 1												
CMU	290	228	14%	104 s	34%	42%	1.2%	36.3%	50%	57%	74%	26 s
3c8	284	235	15%	78 s	25%	37%	4.5%	44.8%	-	52%	80%	24 s
4c8	297	237	10%	84 s	31%	35%	5.4%	48.4%	-	44%	76%	27 s
1c8	318	250	13%	86 s	25%	28%	6.3%	48.2%	-	53%	78%	21 s
3c8+NN6	264	213	15%	92 s	33%	38%	0.4%	41.3%	22%	56%	79%	21 s
1c8+NN12	261	213	14%	100 s	36%	44%	0.4%	13%	46%	56%	75%	28 s
1c8+NN6	288	229	10%	73 s	22%	33%	1%	48.3%	20%	50%	79%	21 s
3c12+NN6	257	212	15%	99 s	31%	37%	0%	27.6%	16%	53%	75%	25 s
1c16+NN6	276	209	11%	97 s	37%	45%	0.4%	15.2%	16%	50%	82%	25 s
2c12+NN6	294	231	13%	86 s	26%	33%	0%	29.6%	20%	50%	66%	23 s
3c8+Xato-cifs	337	256	10%	81 s	25%	34%	3.8%	51.4%	23%	48%	76%	20 s
1c8+Xato-cifs	287	241	14%	70 s	26%	32%	4.5%	46.6%	32%	54%	77%	19 s
1c8+Pwned-fs	292	242	13%	85 s	24%	28%	0%	41.9%	23%	52%	86%	23 s
1c8+Xato-strip-cifs	311	267	12%	93 s	27%	35%	1%	41%	50%	53%	79%	25 s
1c8+Xato-ciss	261	200	26%	139 s	49%	47%	0%	24.4%	78%	65%	67%	23 s
Experiment 2												
CMU	429	333	13%	98 s	33%	42%	1.6%	37.9%	-	56%	81%	24 s
1c8+NN8	381	291	11%	86 s	30%	35%	0.8%	40.2%	27%	50%	72%	24 s
1c8+NN10	385	293	13%	109 s	33%	38%	0.5%	31.7%	34%	52%	80%	24 s
1c10+NN8	381	286	11%	89 s	32%	40%	0.8%	31%	25%	51%	77%	23 s
1c10+NN10	401	303	12%	92 s	32%	41%	0%	25.2%	30%	49%	75%	27 s
1c12+NN10	378	273	14%	95 s	28%	38%	0.3%	19.8%	22%	58%	73%	22 s
1c8+Pwned-fs	435	322	16%	83 s	25%	33%	0.7%	43%	-	47%	75%	25 s
1c8+Xato-strip-cifs	434	327	11%	97 s	29%	35%	2%	40.1%	-	55%	76%	21 s
4c8+Pwned-fs	378	287	16%	90 s	29%	33%	3.1%	50.6%	-	56%	71%	24 s
4c8+Xato-strip-cifs	403	299	14%	99 s	32%	41%	1.4%	41.7%	-	51%	81%	25 s

Table 2: Descriptive statistics for Experiments 1 and 2. Recall time and success rates are for Part 2. Median creation and recall times are shown. "Noticed reject" refers to rejection by a minimum-strength or blocklist requirement.

would provide sufficient protection against 10^6 online attacks, we focused on *NN6* policies and varied composition requirements. All pairwise comparisons were made against the *3c8+NN6* baseline, as our initial retrospective analyses suggested the *3c8+NN6* policy would produce overall stronger passwords than other *NN6* policies. We find that **while minimum-strength policies can be strengthened against offline attacks by either increasing the minimum required length or the minimum number of character classes, increasing the minimum length accomplishes this at a lower usability cost, in terms of how long users need to create a compliant password and how annoying or challenging they find that task**.

As shown in Figure 3, combining a *NN6* minimum-strength requirement with different additional requirements led to passwords that differed substantially in overall guessability (PP). We did not find statistically significant differences in the number of *NN6* passwords guessed for online attacks (i.e., up to 10^6 guesses). However, against a 10^{14} offline attack, policies differed in their defensive effectiveness. While *1c8+NN6* provided similar protection to *3c8+NN6* (48% vs. 41% guessed, respectively), *1c16+NN6* (15% guessed), *2c12+NN6* (30% guessed), and *3c12+NN6* (28% guessed) all offered significantly more protection than *3c8+NN6* (FET).

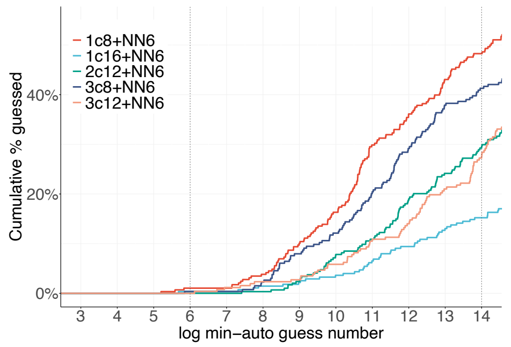


Figure 3: Min-auto guess numbers for policies containing *NN6* requirements and varying composition requirements.

Most of the *NN6* policies we tested showed similar usability properties relative to our *3c8+NN6* baseline; only *1c8+NN6* performed better on two of our usability metrics. Compared to *3c8+NN6* participants, *1c8+NN6* participants reported password-creation to be less difficult (FET: 22% vs. 33% found difficult) and also took less time (MWU: median of 73 s vs. 92 s).

Overall, our results show that, for policies enforcing a particular minimum-strength requirement, more complex composition requirements can lead to passwords that are more resistant to guessing attacks, particularly for offline attack scenarios. While our results show that requiring more character classes or longer passwords both make passwords stronger, we found evidence that increasing the length requirement could produce larger security benefits than increasing character-class-count requirements, while also having less of a negative impact on password-creation usability (e.g., *1c8+NN6* vs. *3c8+NN6*, compared to *3c8+NN6* vs. *3c12+NN6*).

4.4 RQ4: Blocklists vs. min-strength policies

A high-level goal for both experiments was to compare *1c8* policies that incorporated a blocklist requirement to those that instead incorporated a minimum-strength requirement. In Experiment 1, we found that **a *1c8+NN6* minimum-strength policy can provide similar protection against online attacks and similar usability compared to the two best-performing blocklist policies we tested.** In Experiment 2, we compared those two blocklist policies to two additional minimum-strength policies, *1c8+NN8* and *1c8+NN10*. **Both *1c8+NN8* and *1c8+NN10* led to overall stronger passwords than the blocklist policies, while maintaining comparable usability.**

4.4.1 RQ4.A. As shown in Figure 2, neither *1c8+Xato-cifs* nor *1c8+Xato-ciss* resulted in passwords comparable in strength to those created under *1c8+NN6*. Compared to *1c8+NN6* passwords, *1c8+Xato-cifs* passwords were overall significantly weaker (PP). *1c8+Xato-ciss* resulted in passwords that were significantly stronger than those created under *1c8+NN6* (PP), but at the expense of the severe usability issues described in Section 4.2.

Two blocklist policies provided comparable security to *1c8+NN6*, in terms of general guessability as well as resistance to online guessing attacks: *1c8+Pwned-fs* and *1c8+Xato-strip-cifs*. We did not find any statistically significant differences between either blocklist policy and *1c8+NN6* for any of the usability metrics we measured. Thus these three policies appear to be suitable for preventing predictable passwords that might be compromised in online attacks.

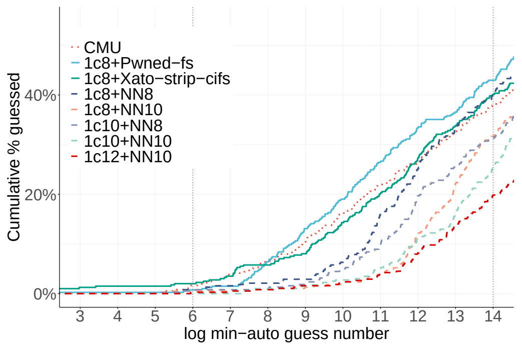


Figure 4: Min-auto guess numbers for Experiment 2 *1c8* and *CMU* policies.

4.4.2 RQ4.B. As shown in Figure 4, we found that both *1c8+NN8* and *1c8+NN10* policies produced passwords that were overall stronger when compared to either of *1c8+Pwned-fs* or *1c8+Xato-strip-cifs* passwords (PP). For attackers making 10^{14} guesses, *1c8+NN8* passwords would be guessed with similar success rates compared to either *1c8+Pwned-fs* or *1c8+Xato-strip-cifs* passwords. In contrast, *1c8+NN10* passwords remained less likely to be guessed, even against the number of guesses possible in offline attacks: 32% of *1c8+NN10* passwords would be guessed within 10^{14} attempts, compared to 40% of *1c8+Xato-strip-cifs* passwords and 43% of *1c8+Pwned-fs* passwords (FET). Across our usability measurements, we did not find statistically significant differences between *1c8+NN8* or *1c8+NN10* and either blocklist policy, except when comparing *1c8+Pwned-fs* to *1c8+NN10*: *1c8+Pwned-fs* participants took less time to create their password than *1c8+NN10* participants (MWU: median of 83 s vs. 109 s) and were less likely to report password-creation as being difficult (FET: 25% vs. 33% found difficult).

Condition	Peto-Peto	percent @ 10^6	percent @ 10^{14}	Creation time	Creation difficult
RQ4.B					
1c8+Pwned-fs		0.7%	43%	83 s	25%
1c8+NN8	p<.001	0.8%	40.2%	86 s	30%
1c8+NN10	p<.001	0.5%	31.7%	109 s	33%
1c8+Xato-strip-cifs		2%	40.1%	97 s	29%
1c8+NN8	p=.009	0.8%	40.2%	86 s	30%
1c8+NN10	p<.001	0.5%	31.7%	109 s	33%
RQ5					
1c10+NN8		0.8%	31%	89 s	32%
1c8+NN8	p=.628	0.8%	40.2%	86 s	30%
1c10+NN10	p=.002	0%	25.2%	92 s	32%
1c12+NN10	p=.925	0.3%	19.8%	95 s	28%
1c10+NN10		0%	25.2%	92 s	32%
1c8+NN10		0.5%	31.7%	109 s	33%
1c8+NN10		0.5%	31.7%	109 s	33%
1c10+NN10	p=.752	0%	25.2%	92 s	32%
1c8+NN8	p<.001	0.8%	40.2%	86 s	30%

Color key	
🟦 Baseline for pairwise comparisons	
🟩 Pairwise test statistically significant (better than baseline)	
🟥 Pairwise test statistically significant (worse than baseline)	
⬜ Pairwise test not stat. sig.	▨ Omnibus test not stat. sig.

Table 3: RQ4.B and RQ5 strength/usability comparison. Pairwise comparisons for non-baseline policies are for the baseline policy listed immediately above.

These results demonstrate the value of minimum-strength requirements. While both blocklist and minimum-strength requirements can prevent users from picking common passwords that attackers are likely to try first, minimum-strength requirements can do so while also making passwords harder to guess for more determined attackers (see top half of Table 3).

4.5 RQ5: Min-strength and min-length requirement interactions

We tested additional minimum-strength policies with the goal of identifying configurations with more positive password-creation usability properties yet similarly strong offline attack protections as minimum-strength policies we had tested in Experiment 1. We focused our exploration of policies on those that only enforced minimum-strength and minimum-length requirements, without any character-class requirements. **Our results ultimately show that *NN10* requirements can provide stronger protection against offline attacks than *NN8* requirements without introducing substantial usability harm, and that combining *NN10* requirements with a minimum length of 10 characters can help users create passwords more quickly.**

Unsurprisingly, increasing the minimum-strength requirement while fixing the minimum password length was effective at increasing overall password strength (PP: *1c8+NN8* vs. *1c8+NN10*, *1c10+NN8* vs. *1c10+NN10*). We find evidence that increasing a policy's minimum-strength threshold from *NN8* to *NN10* also strengthened offline attack defenses, as shown in Figure 4. For example, 40% of *1c8+NN8* passwords would be guessed in a 10^{14} attacks compared to only 32% of *1c8+NN10* passwords, a difference which was statistically significant (FET). More interestingly, for the min-length-8 and min-length-10 policies we explored, participants did not find passwords substantially more difficult or annoying to create regardless of whether their policy included a *NN8* or *NN10* requirement. The only statistically significant usability difference we found between *NN8* and *NN10* policies was with respect to password-creation time: *1c8+NN8* passwords took slightly less time to create than *1c8+NN10* passwords (MWU: median of 86 s vs. 109 s).

We performed similar comparisons between minimum-strength policies with the same required strength threshold but varying minimum-length requirements. For policies enforcing the same minimum-strength requirement, increasing the minimum-length requirement tended to produce stronger passwords. As shown in Figure 4, for policies containing *NN8* or *NN10* minimum-strength requirements, passwords were less likely to be guessed in a 10^{14} offline attack if that policy imposed *1c10* rather than *1c8* requirements (FET). We did not find large differences in reported password-creation annoyance or difficulty between *NN8* and *NN10* policies depending on whether those policies required passwords to be at least 10 or 12 characters. While we find some support for our initial hypothesis that minimum-length requirements can make minimum-strength requirements easier to satisfy, the improvements were small (e.g., *1c10+NN10* passwords were created in 17 fewer seconds than *1c8+NN10* passwords, on average). Interestingly, only 22% of participants assigned to the *1c12+NN10* policy reported noticing a password they wanted to create being rejected by the minimum-strength requirement, a significantly lower percentage than we found for participants assigned to either *1c10+NN10* (FET: 30% noticed) or *1c8+NN10* (FET: 34% noticed). However, differences in proportions of participants who noticed a minimum-strength rejection did not translate to statistically significant differences in reported password-creation annoyance or difficulty (Figure 11). Table 3 (bottom half) summarizes these results.

Overall, we found little reason to prefer *NN8* policies over the stronger and similarly usable *NN10* policies. Among *NN10* policies, *1c12+NN10* appeared particularly attractive. Compared to *1c8+NN10* and *1c10+NN10*, *1c12+NN10* led to participants encountering minimum-strength rejections less often and improved resistance to offline guessing attacks.

4.6 RQ6: Blocklist and composition requirement interactions

We compared *1c8+Xato-strip-cifs*, *1c8+Pwned-fs*, *4c8+Xato-strip-cifs*, and *4c8+Pwned-fs* with each other in order to understand how blocklist requirements interacted with character-class requirements. Our results confirm that the choice of blocklist can impact usability and security differently depending on whether it is included in a *1c8* or *4c8* policy. **While the choice of *1c8* or *4c8* composition requirements to combine with a given blocklist did not significantly impact usability, we found evidence that *4c8* requirements could *negatively* impact security relative to *1c8* requirements if combined with a very large blocklist and full-string matching.**

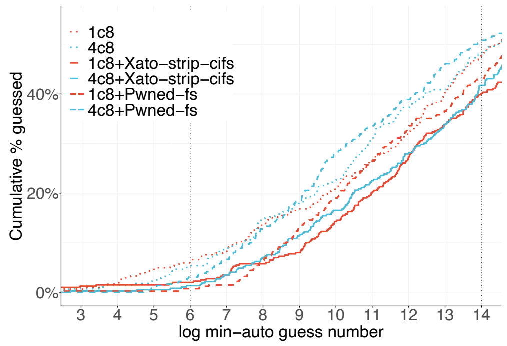


Figure 5: Blocklist/composition policy interaction effects. *1c8* and *4c8* curves are based on data collected from Experiment 1 and are shown only for reference.

As in Experiment 1, passwords created under *1c8+Pwned-fs* were not statistically significantly different in their guess-number distribution than those created under *1c8+Xato-strip-cifs*. However, we did find usability differences: *1c8+Pwned-fs* participants were more likely to drop out of our study before creating their password than *1c8+Xato-strip-cifs* participants (FET: 16% vs. 11% dropped out) and those who did not drop out took slightly longer to create their passwords (MWU: median of 97 s vs. 83 s).

We performed similar comparisons between our *4c8+Pwned-fs* and *4c8+Xato-strip-cifs* conditions. Unlike the case for *1c8* requirements, we found that *4c8+Pwned-fs* passwords were overall weaker than *4c8+Xato-strip-cifs* passwords (PP). Although this only translated to substantially higher likelihood of passwords being guessed for 10^{14} offline attacks (FET: 51% vs 42%) and not 10^6 online attacks, Figure 5 suggests *4c8+Xato-strip-cifs* also may be preferable to *4c8+Pwned-fs* for online attack scenarios. In terms of usability, we found that—unlike for *1c8* policies—a higher proportion of participants were annoyed by *4c8+Xato-strip-cifs* compared to *4c8+Pwned-fs* (FET: 41% vs 33% annoyed).

We found support for our hypothesis that blocklists of large password leaks are more effective when combined with *1c8* requirements than with *4c8* requirements. Passwords created under *4c8+Pwned-fs* were statistically significantly weaker than *1c8+Pwned-fs* passwords, both overall (PP) and in terms of resistance to 10^6 (FET: 3.1% vs. 0.7% guessed) and 10^{14} guessing attacks (FET: 50.6% vs. 43%

guessed). We found less support for our hypothesis that *Xato-strip-cifs* makes password creation a more frustrating experience when combined with *4c8* requirements than with *1c8* requirements.

5 DISCUSSION

5.1 Character-class requirements

Although prior work has repeatedly found that requiring more character classes decreases guessability [11, 16], researchers have shown that character-class requirements lead to frustration and difficulty for users [13, 23, 31]. Since other requirements, e.g., minimum-length or blocklist requirements, can strengthen passwords with less negative impact on usability research has advocated retiring character-class requirements [7, 26]. These recommendations have been standardized in recent NIST password-policy guidance [19].

Our experimental results provide the first concrete evidence that character-class requirements should be avoided not only because users tend to find them annoying, but also because they don't provide substantial benefit against attackers using state-of-the-art password-cracking tools: an expert attacker can guess *1c8*, *3c8*, and *4c8* passwords with equal success rates.[7] Experiment 2 also suggests that character-class requirements should be avoided for password-creation policies that include a blocklist or minimum-strength requirement. We find evidence that policies requiring all four character classes and a large blocklist check produce passwords that are, at best, as strong as passwords created under a policy that performed the same blocklist check without character-class requirements. Although Experiment 2 does show that a policy requiring more character classes does tend to produce stronger passwords under a minimum-strength requirement, it also shows that this strength improvement is much lower than the improvement that results from increasing either length requirements or minimum-strength threshold requirements.

Our new findings on character-class requirements seem to be caused by two factors. First, users tend to create passwords that are longer and contain more character classes than required; this was more pronounced in our experiments than in previous studies. In Appendix C, we examine passwords from studies conducted from 2010 to 2019 and observe that in more recent studies, and in those with a password meter, users were more likely to exceed length and character-class requirements. Second, password guessing has improved over time, and more so for passwords containing three or four character classes than for passwords containing one or two classes. We describe this in more detail in Appendix D.

5.2 Blocklist requirements

Blocklists can be useful, but only if carefully configured. Both the wordlist and matching algorithm can significantly affect password strength and usability. Our experiments show that some blocklist configurations are much more likely than others to make password creation a frustrating and time-consuming experience. Given a blocklist composed of passwords that appear at least four times in public password leaks, a blocklist policy configured to reject passwords that contain a five-character substring of any blocklisted password will strengthen resulting passwords, but with a severe impact on usability. We also find blocklist configurations that do not offer sufficient strength protections. Full-string matching against a list of roughly 10^5 commonly leaked passwords leads to a negligible improvement in defense against guessing attacks. However, a more fuzzy matching algorithm such as one that performs the same check ignoring non-alphabetic characters can improve password strength without requiring a larger set of leaked passwords to check against.

We recommend that policies containing blocklist requirements should not additionally require passwords to contain a minimum number of character classes, especially if the blocklist check is based on rejecting any password exactly matching one that has been previously compromised in a public leak. Although further research is needed to confirm this, our results suggests that passwords subjected to such a blocklist check may actually be weaker if additionally required to contain all four character classes. One explanation for this is that easily-guessed *4c8* passwords may be less likely to appear in public leaks than easily-guessed *1c8* passwords and so are less likely to be included in blocklists.

Two *1c8* blocklist configurations we tested performed well with respect to both security and usability: the *Xato* wordlist combined with *strip-cifs* matching and the *Pwned* blocklist combined with *fs* matching. The *Pwned-fs* blocklist configuration may be stronger against online attacks than the *Xato-strip-cifs* blocklist. Although the two configurations have similar usability in general, the higher dropout rate in Experiment 2 for *1c8+Pwned-fs* suggests that that policy may be more frustrating for some users. Unlike the easily-embedded *Xato-strip-cifs* blocklist, the *Pwned-fs* blocklist is less useful for fully client-side password checking.

System administrators incorporating a blocklist check should check for password-policy compliance remotely on the server, since clients collecting passwords can misbehave or lie. Password checks against externally controlled databases of leaked passwords involve some security risk, even if mitigated by transmitting partial password hashes. Concerned organizations can perform leaked-password checks locally, either against a large set of leaked passwords (30 GB uncompressed for a comprehensive set [10]) or against a Bloom filter, which substantially reduce storage space requirements at the expense of false positives (1.1 GB for a Bloom filter with a false positive rate of 0.1% for the same set). If using a large blocklist with a Bloom filter or a smaller blocklist with a *strip-cifs* matching, it may be useful to also check for policy compliance locally to facilitate real-time requirements feedback without network latency introduced by remote checks.

5.3 Minimum-strength requirements

Our results confirm that minimum-strength requirements can effectively guide users toward stronger passwords without significantly inhibiting password memorability or ease of password creation. The choice of minimum-strength threshold depends on security requirements; too low a threshold may not provide enough defense (particularly against online attacks) and too high a threshold may unacceptably inhibit usability.

When online attacks are a concern, we recommend setting the minimum-strength threshold to at least 10^6. We recommend using minimum-strength requirements alongside minimum-length

[7]We assume passwords are created with feedback that includes a strength meter and text feedback, similar to our study.

requirements, but without blocklists or character-class requirements. While our results suggest that, when combined with a minimum-strength requirement, policies that require multiple character classes improve resistance against offline attacks, similar improvement can be achieved with less impact on password-creation usability by increasing the minimum-strength threshold or the minimum required length. For example, *1c8+NN12* had similar usability to *3c8+NN6* but led to much stronger passwords (post-hoc comparison, see Table 6 in Appendix).

Increasing the minimum-strength threshold improves the security of passwords, but too high a threshold can make creating passwords difficult and annoying. For example, for online services that prioritize a seamless user experience, usability impacts from a *1c8+NN12* policy may be unacceptable.

One might question why minimum-strength thresholds above 10^6 guesses but below 10^{14} guesses are useful; a 10^6 threshold will help avoid the most predictable passwords that might be guessed in an online attack, and in scenarios where attackers can make more than 10^6 guesses, any threshold below 10^{14} will not be enough to prevent account compromise in an offline guessing attack on a hashed and salted database of passwords [7]. Although resistance to guessing attacks between feasible online (10^6) and extensive offline (10^{14}) attacks may not always prevent account compromise, we note that policies with minimum-strength thresholds between 10^6 and 10^{14} resulted in significantly higher resistance to offline guessing attacks at a 10^{14} cutoff; a *NN8* minimum-strength policy not only prevents the most predictable 10^8 passwords, but also increases the proportion of created passwords with guess numbers above 10^{14} relative a *NN6* policy. In addition, the use of tunable, slow hashing algorithms and increased computational resources for guessing attacks over time means that guessing thresholds other than 10^6 and 10^{14} (including 10^8 and 10^{10}) can be relevant to protect against. As demonstrated in Experiment 2, by combining minimum-strength requirements with specific minimum-length requirements, we can achieve this benefit of increased resistance to offline guessing without incurring substantial negative impact on the ease of password creation.

In general, increasing length requirements for a given minimum-strength policy strengthened passwords produced under that policy against offline attacks, while maintaining strength against online attacks. Since we didn't find any significant usability differences between length variants of the *1c8* and *1c10* minimum-strength policies that we tested, this suggests that the longer-length versions of these policies should be preferred for their security benefits. However, we note that longer minimum-length requirements for a given minimum-strength requirement do not always come without a usability hit; in Experiment 1, *1c16+NN6* led to noticeably higher levels of reported annoyance and difficulty during password creation than did *1c8+NN6*. This illustrates the importance of testing specific combinations of minimum-length and minimum-strength requirements before their deployment in a password policy.

Synthesizing these results, we recommend a password policy of *1c12+NN10* for security settings that need protection against offline attacks while still providing reasonable usability. Similar to blocklists, system administrators using a minimum-strength requirement should check that the requirement is met at both the client and at the remote server.

5.4 Blocklists vs. minimum-strength policies

A main goal of our study was to compare blocklist and minimum-strength policies while considering both security and usability. We find that both types of policies can protect well against 10^6 online attacks without unduly harming usability: both the *1c8+Xato-strip-cifs* and *1c8+Pwned-fs* blocklist policies achieve these goals, as does the minimum-strength *1c8+NN6* policy. However, we also find that minimum-strength policies can provide better protection than top-performing blocklist policies against offline attacks, while improving usability.

Why use minimum-strength policies if blocklists are adequate as a defense against online attacks and easy to deploy? Prior work has also suggested that protecting against guessing attacks beyond the threshold of effort possible in online attacks may be wasted effort, e.g., users suffer through a more painful password-creation process for no practical security gain [7]. We believe minimum-strength policies are a good alternative, for a few reasons. First, we believe minimum-strength policies can be deployed just as easily—if not more easily—than blocklist policies, particularly as the smaller blocklists we tested were not adequately effective. Our results are based on a client-side NN model that could be pre-trained and distributed to system administrators. Second, users often reuse passwords, including between high-value and low-value accounts [22, 32]. Even if an organization is not concerned about offline attacks, e.g., due to sophisticated hashing, its users may reuse their passwords on less secure systems. In this case, resistance to offline attacks would still be relevant. Third, we showed that using minimum-strength policies can increase password strength without noticeable negative impact on user experience.

6 CONCLUSION

We explored in depth three types of requirements enforced in password policies: composition requirements, blocklists, and neural-network-driven minimum-strength requirements. Using two large-scale, experimentally designed user studies, we examined the security and usability of each type of requirement, and their combinations, when deployed in a modern password meter. Our results lead to concrete recommendations for configuring blocklist requirements. We recommend that blocklist requirements either check candidate passwords against a list of about 10^5 commonly leaked passwords using a fuzzy matching algorithm or perform a full-string check against a large list consisting of all known leaked passwords. Password policies incorporating blocklist requirements should not impose character-class requirements. We also find that minimum-strength policies, which we believe our work to be the first to closely investigate in a user study, can improve upon blocklist policies by increasing resilience to offline attacks without degrading usability. We recommend a *1c12+NN10* minimum-strength policy for organizations that wish to protect high-value accounts without a substantial negative usability impact.

ACKNOWLEDGMENTS

The authors thank Billy Melicher, Blase Ur, Maggie Oates, Hana Habib, Kentrell Owens, our shepherd Daniel Zappala, and the anonymous reviewers. This project was funded in part by gifts from Microsoft Research, Google, and NVIDIA.

A NEURAL NETWORK TRAINING

Here we present additional tables and plots related to the training and evaluation of the PGS3 NN models used in our user studies.

Prior to conducting our user study, we trained neural network models for predicting password strength. Our models were created using code based on prior work by Melicher et al. [16]. Similar to their work, we trained a large Tensorflow-based Keras model containing three recurrent layers and two densely connected layers. Each recurrent layer contained 512 LSTM units and each dense layer contained 1,000 units. Our models used a vocabulary of 96 characters, including lowercase, uppercase, digits, and symbols. The minimum and maximum lengths of passwords predicted by our model were 8 and 30 characters, and the context length was 10 characters wide. We did not apply rare character or lettercase optimizations, nor dropout. Our base *1c8* model was trained using two NVIDIA Tesla P100 GPUs and a batch size of 1024 samples for 20 epochs over a period of 11 days. For each large Keras model, we also trained a smaller version containing 200 units in both the LSTM and dense network layers. We then created a model that could be included in a client-side password meter by converting the smaller Keras model into a TensorflowJS model. While the size of the large NN models were around 100MB, each TensorflowJS NN model was approximately 4.7MB uncompressed, including the precomputed probability-to-guess-number mapping file.

Policy	PGS++ (old)	PGS3 (new)
1c8	73.4 million	32.8 million
1c16	2.5 million	1.5 million
2c12	13.3 million	5.4 million
3c8	13.6 million	5.7 million
3c12	4.1 million	1.8 million
4c8	311 thousand	599 thousand

Table 4: Training data used for PGS3 and PGS++ [16] NN models. Each PGS3 model other than the *1c8* model was trained using transfer learning, initialized according to model weights for a superset policy (e.g., the *3c8* model was trained starting from the trained *1c8* model).

We experimented with transfer learning in order to train policy-specific NN models, which prior work had found improved *1c16* guessing performance, particularly for large guess numbers [16].[8]

A primary difference between the PGS++ NN models used in prior work [16] and our PGS3 models is that our models contained a character-level embedding layer. The embedding size was set to eight dimensions. Another difference between the prior PGS++ models and our PGS3 models is the training data used for each set of models, which consisted of of PGS-compliant passwords in the LinkedIn, Mate1, RockYou, and 000webhost datasets. As shown in Table 4, our training data contains many fewer *1c8* passwords, but slightly more *4c8* passwords.

We compared the guessing effectiveness of our PGS3 NN models and the prior PGS++ model for *1c8*, *3c8*, and *4c8* passwords collected in Experiment 1. As demonstrated in Figure 6, the PGS3 models guessed passwords from each of these policies more effectively than

[8]We only trained composition-requirements-specific NN models for Experiment 1. These models were also used for Experiment 2.

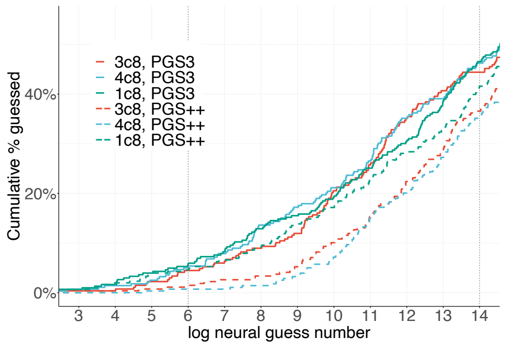


Figure 6: Guessing performance of the PGS++ and PGS3 NN models for passwords collected in Experiment 1. In contrast to the previous PGS++ model, the PGS3 model guesses *1c8*, *3c8*, and *4c8* passwords at similar rates.

the PGS++ model. Furthermore, we find that the improvement in guessing performance is largest for *4c8*, followed by *3c8*, and then by *1c8*. While we do notice a small increase in guessing performance for *1c8* passwords, this increase was not statistically significant. For *3c8* and *4c8* passwords, the increase was statistically significant (PP; *3c8*: χ=11.9, p=.001; *4c8*: χ=18.2, p<.001). We experimented with many model variations in order to understand which differences between the PGS3 and PGS++ models led to these results. We found that the majority of the guessing improvement of PGS3 over PGS++ could be attributed to the inclusion of the embedding layer in the PGS3 model. On the other hand, the PGS3 training data and the application of policy-specific transfer learning produced only slight improvements in guessing performance.

B MINIMUM-STRENGTH-REQUIREMENT ATTACKERS

For a minimum-strength policy, the underlying NN used to estimate password strength and allow or reject passwords can be leveraged by attackers to improve guessing effectiveness. As a rough analysis of how much benefit this kind of attacker knowledge provides, we plot and compare guess curves for both situations. To simplify analysis, we plot the password guessability under each policy according to browser-NN guess numbers. These guess numbers are similar to PGS3-NN guess numbers that we primarily report, modulo artifacts introduced during NN compression.

As illustrated in Figure 7, a minimum-strength-aware attacker does gain a noticeable increase in guessing effectiveness, equivalent to those passwords able to be guessed in the head start afforded by skipping minimum-strength-rejected passwords (e.g., skipping 10^6 passwords for a *NN6* policy). However, this benefit quickly fades within an order of magnitude or fewer guesses.

C HOW HAVE PASSWORDS CHANGED?

Results from our experiments related to the impact of character-class requirements on password guessability differed from those reported in prior work. Prior work has found that policies requiring more character classes tend to produce overall stronger passwords. In contrast, in our study participants assigned to *1c8*, *3c8*,

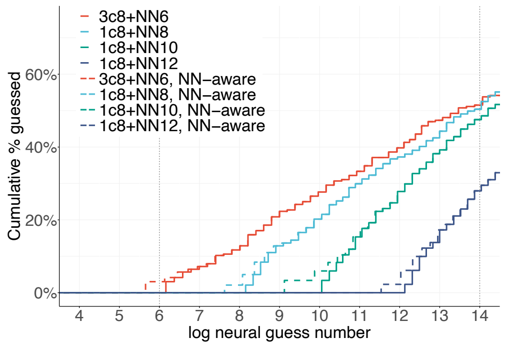


Figure 7: Password guessability against minimum-strength-aware and minimum-strength-oblivious attackers. Both types of attackers order password guesses from most probable to least probable. The minimum-strength-aware attacker additionally avoids making guesses that the minimum-strength requirement would have rejected. The *NN8* and *NN10* conditions are from Experiment 2. This plot shows browser-NN guess numbers.

and *4c8* policies created passwords that did not significantly differ in strength. As noted in Section 5.1, this change appeared to be in part due to the fact that participants in our study were more likely to exceed composition requirements than participants from prior work. To investigate further, we compared how password lengths and character-class compositions have changed across both prior work and in our study.

We focused our comparisons to passwords collected in three separate studies under a *1c8* policy with no other restrictions, which used a similar methodology to ours (i.e., Mechanical Turk participants and use of role playing to create an email account password) [11, 28, 29]. Although these passwords were collected under similar methods to ours, some differences exist. We identified primary differences that could have potentially impacted participants' passwords, which included when data was collected and the password-creation interface that participants created passwords under. We attempted to understand how these differences related to our new findings by comparing the character-class composition and length of passwords collected in each study.

	Without meter			**With meter**	
	2010	**2012**	**2016**	**2016**	**2019**
One class	38%	14%	11%	7%	4%
Two classes	46%	20%	39%	26%	18%
Three classes	12%	49%	32%	35%	30%
Four classes	4%	16%	19%	32%	48%

Table 5: Number of character classes contained in *1c8* passwords collected in studies that were conducted in 2010 [13], 2012 [29], 2016 [28], and 2019 (our study).

We find that more passwords contain multiple character classes over time. Part of this appears to be due to the password meter used to collect passwords in each study. Comparing 2016 study passwords collected without a meter to passwords from the same study collected using a meter (similar to ours), we see a 13% jump in the percentage of passwords containing four classes. We also find increases in the number of character classes contained in passwords over time that are not attributable to use of a meter. As shown in Table 5, 2016 passwords collected without a meter had 27% fewer passwords containing exactly one class and 15% more passwords containing four classes compared to 2010 passwords collected without a meter. Similarly, *1c8* passwords collected in our study contained 16% more passwords containing four classes compared to passwords collected in 2016 using the same meter.

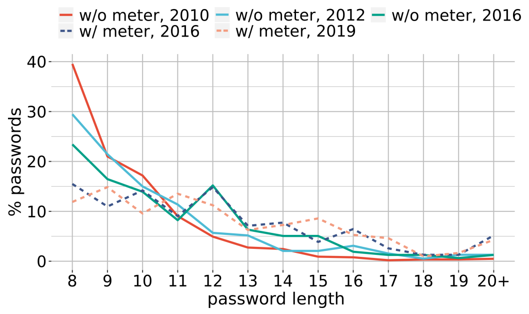


Figure 8: Length distribution for *1c8* passwords collected in studies over time.

Besides containing more character classes, we find that passwords have become longer over time. As illustrated in Figure 8, both the year of the study and the use of a password meter appear to be associated with this change. For example, *1c8* passwords collected without a meter in 2016 had roughly 15% fewer passwords that were exactly eight characters long compared to passwords collected without a meter in 2010. We also see evidence that the password meter played a role in lengthening passwords; 2016 passwords collected with a meter had approximately 10% fewer passwords that were exactly eight characters long compared to passwords collected in the same study without a meter. We observe similar password-length distributions for passwords collected in 2016 and 2019 using the same password meter.

D CHARACTER CLASSES AND GUESSABILITY

In our study we find that character-class requirements are not only annoying, but provide little security benefit in terms of defending against guessing attacks. As mentioned in Appendix C, this result applies to character-class requirements in a policy deployed using a password meter similar to the one we use. Here we show support that a rough relationship still exists between the number of character classes actually contained in a password and password guessability, for passwords collected via a *1c8* policy.

In Figure 9 we plot guess curves for passwords containing an exact number of character classes.[9] We see that the PGS3 NN guesses

[9]We chose to subset passwords created under different policies, rather than subset only *1c8* passwords, in order to focus on passwords created by participants who may tend to satisfy minimum requirements only. If we had performed a similar analysis on only *1c8* passwords, then comparisons between passwords containing exactly one class versus four classes might also implicitly compare passwords created by participants who naturally put different levels of effort into creating strong passwords.

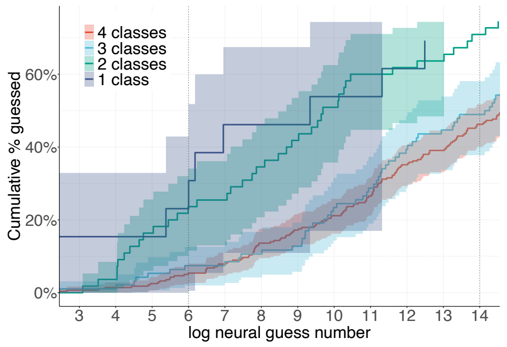


Figure 9: PGS3 NN guess numbers for passwords containing exactly one, two, three, and four class(es), subsetted from passwords created under *1c8*, *3c8*, and *4c8* policies in Experiment 1. Each of these sets contains differing number of passwords; we plot confidence intervals to reflect how this impacts guess curve estimations. The 95% confidence interval for passwords containing exactly one class is large due to the small number of *1c8* passwords that contained exactly one class.

passwords containing exactly three characters or four characters at around the same rate, but guesses passwords containing exactly one or two classes at a much faster rate. In other words, we find evidence that passwords containing three or four character classes tend to be stronger than those containing one or two classes. Similarly, in Experiment 1 we found that character-class requirements, when combined with minimum-strength requirements, do impact resulting password guessability.

Nonetheless, we still recommend avoiding character-class requirements. Users tend to find these requirements annoying and, given an effectively designed password meter, many will incorporate multiple classes anyway of their own accord. Other types of policy requirements, including minimum-strength and minimum-length requirements, annoy users less, have a larger potential effect on resulting password guessability, and can be configured more granularly to achieve a desired security level.

E STUDY INSTRUCTIONS AND SURVEY PROTOCOLS

E.1 User study instructions

f0. MTurk HIT instructions
Please visit the study link below to continue this HIT. After completing the study, you will be provided a completion code, which will be required in order to receive payment. *Make sure to leave this window open as you complete the study.* When you are finished, return to this page to paste the completion code into the box.
[text box for completion code]

1. Participant consent
[consent form]

2. Part 1 instructions
In this study you will be asked to create a password and fill out a 3-minute survey for a 55 cent payment. We will contact you to come back a few days later to try to use your password to log in again and fill out another survey for a 70 cent bonus payment.
Imagine that an online *account that you care a lot about*, such as your main email account, is requiring that all users change their passwords. We will ask you to use this password in a few days to log in again, so it is important that you remember your new password. Please take the steps you would normally take to create and remember your important passwords, and protect this password as you normally would protect your important passwords. Please behave as you would if this was your real password! *NOTE: This password is only being used for the purpose of this study and you should not use a password that you use for your actual email account.*

3. Password-creation task
[password meter used to create password]

4. Part 1 password-recall task
[password entry form]

5. Part 1 survey
[see Appendix E.2]

6. Instructions after completing Part 1 survey
Thank you! You have finished Part 1 of the study. Please enter the following completion code on MTurk: [study completion code]
Please expect an email a few days from now to return for Part 2 of the study, for which you will receive an additional 70 cent bonus payment.

7. Part 2 password-recall task
[password entry form]

8. Part 2 survey
[see Appendix E.3]

9. Instructions after completing Part 2 survey
Thank you! You have finished Part 2 of this study about the memorability and security of passwords. Please expect a 70 cent bonus payment on Mechanical Turk within the next few days.

E.2 Part 1 survey

Thank you for creating a password. Next, we would like you to answer some survey questions.

Creating a password during this study was annoying
◯ [Strongly disagree to Strongly agree] (5-point scale)

Creating a password during this study was fun
◯ [Strongly disagree to Strongly agree] (5-point scale)

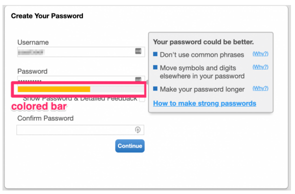
Creating a password during this study was difficult
◯ [Strongly disagree to Strongly agree] (5-point scale)

Please explain why creating a password during this study was/was not difficult.
[text box input]

Which of the following best describes your approach to creating your password in this study?
◯ I reused a password that I currently use, or previously have used, for a different account.
◯ I modified a password that I currently use, or previously have used, for a different account.
◯ I created an entirely new password, but I used the same general approach that I normally do.
◯ I created an entirely new password, and I used a different approach than I normally do.

In general, I am confident in my ability to create strong passwords.
◯ [Strongly disagree to Strongly agree] (5-point scale)

The following questions refer only to the colored bar that measures the strength of your password (highlighted inside pink box).


Create Your Password
Username
[redacted]
Password
Show Password & Detailed Feedback
Confirm Password
Your password could be better.
■ Don't use common phrases (Why?)
■ Move symbols and digits elsewhere in your password (Why?)
■ Make your password longer (Why?)
How to make strong passwords

The colored bar helped me create a stronger password.
◯ [Strongly disagree to Strongly agree] (5-point scale)

The colored bar was informative.
◯ [Strongly disagree to Strongly agree] (5-point scale)

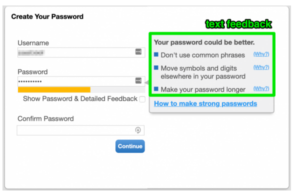
Because of the colored bar, I created a different password than I would have otherwise.
◯ [Strongly disagree to Strongly agree] (5-point scale)

It's important to me that the colored bar gives my password a high score.
◯ [Strongly disagree to Strongly agree] (5-point scale)

What is your opinion of the accuracy of the colored bar's rating?
◯ The colored bar's rating accurately reflected the strength of my password.
◯ The colored bar's rating did not accurately reflect the strength of my password; the colored bar gave my password a lower score than it deserved.
◯ The colored bar's rating did not accurately reflect the strength of my password; the colored bar gave my password a higher score than it deserved.
◯ I don't remember how the colored bar rated my password.

Do you have any other thoughts about the colored bar?
[text box input]

The following questions refer only to the text feedback that measures the strength of your password (highlighted inside green box).


Create Your Password
Username
[redacted]
Password
•••••••••••
Show Password & Detailed Feedback
Confirm Password
Your password could be better.
■ Don't use common phrases (Why?)
■ Move symbols and digits elsewhere in your password (Why?)
■ Make your password longer (Why?)
How to make strong passwords

The text feedback helped me create a stronger password.
◯ [Strongly disagree to Strongly agree] (5-point scale)

The text feedback was informative.
◯ [Strongly disagree to Strongly agree] (5-point scale)

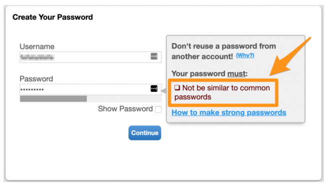
Because of the text feedback, I created a different password than I would have otherwise.
◯ [Strongly disagree to Strongly agree] (5-point scale)

It's important to me that I follow the suggested changes to my password provided by the text feedback.
◯ [Strongly disagree to Strongly agree] (5-point scale)

What is your opinion of the appropriateness of the text feedback?
◯ The text feedback was appropriate for my password; the suggested changes improved the strength of my password.
◯ The text feedback was not appropriate for my password; the suggested changes had no effect on the strength of my password.
◯ The text feedback was not appropriate for my password; the suggested changes weakened the strength of my password.
◯ I don't remember whether the text feedback was appropriate for my password.

Do you have any other thoughts about the text feedback?
[text box input]

[if assigned to a minimum-strength condition]
The following feedback was shown if you attempted to create a common password:



Create Your Password

Username

Password

Show Password

Continue

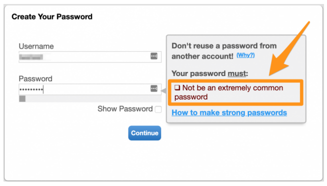
Don't reuse a password from another account! (Why?)

Your password must:

❑ Not be similar to common passwords

How to make strong passwords

[if assigned to a Xato-based blocklist condition]
The following feedback was shown if you attempted to create a common password:



Create Your Password

Username

Password

Show Password

Continue

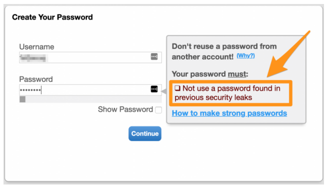
Don't reuse a password from another account! (Why?)

Your password must:

❑ Not be an extremely common password

How to make strong passwords

[if assigned to the Pwned blocklist condition]
The following feedback was shown if you attempted to create a password found in previous security leaks:



Create Your Password

Username

Password

Show Password

Continue

Don't reuse a password from another account! (Why?)

Your password must:

❑ Not use a password found in previous security leaks

How to make strong passwords

[if assigned to a minimum-strength or blocklist condition]
Were any passwords you tried to create rejected for the specific reason shown above (highlighted in orange)?
◯ Yes, I remember these passwords were rejected for that specific reason (list rejected passwords in box)
[optional text box input]
◯ Yes, but I don't remember which passwords were rejected / ◯ No / ◯ I don't remember

With what gender do you identify?
◯ Male / ◯ Female / ◯ Other / ◯ I prefer not to answer

How old are you?
[numeric text box input]

Are you majoring in or do you have a degree or job in computer science, computer engineering, information technology, or a related field?
◯ Yes / ◯ No / ◯ I prefer not to answer

E.3 Part 2 survey

Which of the following statements best reflects *how you entered your password on the previous screen?*
◯ My password was automatically entered for me by a password manager or by my browser
◯ I typed my password in entirely from memory
◯ I had written my password down on paper, and I typed it in after looking it up
◯ I had saved my password electronically (e.g., in a file or on my phone), and I typed it in after looking it up
◯ I had written down or electronically stored a hint (not the password itself) to help me remember my password for this study, and I typed the password in after looking at the hint
◯ Other [text box input]

It was difficult for me to remember the password I entered on the previous screen
◯ [Strongly disagree to Strongly agree] (5-point scale) / ◯ I did not recall the password from memory

Which of the following statements reflect *how you normally enter passwords in your daily life?* (Choose all that apply)
☐ My passwords are automatically entered for me by a password manager or by my browser
☐ I type my passwords in entirely from memory
☐ I write my passwords down on paper, and I type them in after looking them up
☐ I save my passwords electronically (e.g., in a file or on my phone), and I type them in after looking them up
☐ I write down or electronically store hints to help me remember my passwords, and I type my passwords in after looking at those hints
☐ Other
[text box input]

Regardless of how you entered your password on the previous screen, did you do any of the following after you created your password? (Choose all that apply)
☐ I stored my password for this study in a password manager or in my browser
☐ I wrote my password for this study down on paper
☐ I took steps to memorize my password
☐ I stored my password for this study electronically (e.g., in a file or on my phone)
☐ I wrote down or electronically stored hints to help me remember my password for this study, but not my password itself
☐ I did not do any of the above
☐ Other [text box input]

What would you have done differently in creating, protecting, and remembering your password if this password were used for an account you use outside this study?
[text box input]

Do you use the password you created for this study for any other account?
◯ Yes / ◯ No / ◯ I prefer not to answer

F ADDITIONAL DETAILS OF EXPERIMENTAL RESULTS

Figures 10 and 11 show participants' annoyance with and perceived difficulty of password-creation.

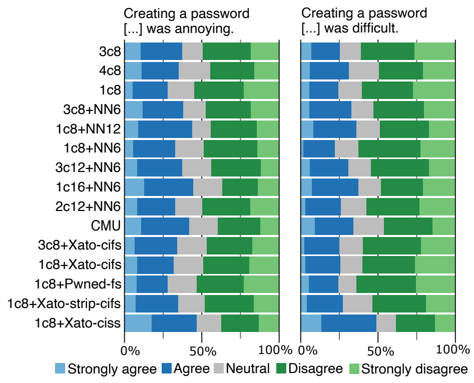


Figure 10: Experiment 1: Participants' level of agreement with the statement above each chart.

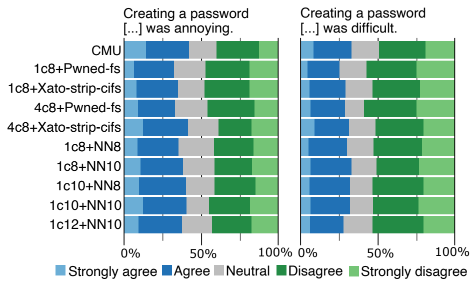


Figure 11: Experiment 2: Participants' level of agreement with the statement above each chart.

F.1 Strength/usability summary tables

Tables 6 and 7 show the results of omnibus and pairwise statistical tests for both experiments. Table 3 explains how to read these tables. We report Holm-Bonferroni-adjusted p-values. See Table 2 for descriptive statistics explaining effect sizes.

Condition	Peto-Peto	percent @ 10^{6}	percent @ 10^{14}	Noticed rejection	Creation time	Creation difficult	Creation annoying	Stored password	Part 1 dropout	Part 2 dropout	Recall time	Recall success	Remember difficult
Impact of blocklists													
3c8													
3c8+Xato-cifs	p=.842	p=.835	p=.114	N/A	p=.895	p=.926	p=.449	p=.941	p=.07	p=.047	p=.101	p=.524	p=.176
1c8												p=.076	
1c8+Xato-cifs	p=.54	p=.46	p=.737	N/A	p=.114	p>.999	p=.574	p>.999	p>.999	p=.29	p=.57		p>.999
1c8+Pwned-fs	p=.54	p<.001	p=.267	N/A	p=.722	p>.999	p>.999	p>.999	p>.999	p=.436	p=.418		p=.645
1c8+Xato-strip-cifs	p=.54	p=.001	p=.25	N/A	p=.558	p>.999	p=.215	p>.999	p>.999	p=.086	p=.29		p>.999
1c8+Xato-ciss	p=.002	p<.001	p<.001	N/A	p<.001	p<.001	p<.001	p=.038	p<.001	p=.616	p=.57		p=.931
Blocklist requirements for *1c8* policies													
1c8+Xato-strip-cifs													p=.064
1c8+Xato-cifs	p=.017	p=.048	p=.399	p=.051	p=.003	p=.808	p=.544	p>.999	p=.988	p=.738	p=.016	p=.745	
1c8+Xato-ciss	p=.013	p=.498	p<.001	p<.001	p<.001	p<.001	p=.011	p=.031	p<.001	p=.015	p>.999	p=.31	
1c8+Pwned-fs	p=.8	p=.498	p=.865	p<.001	p=.142	p=.808	p=.161	p>.999	p=.988	p=.738	p>.999	p=.332	
Composition requirements for minimum-strength policies													
3c8+NN6		p=.224		p=.436				p=.705	p=.272	p=.385	p=.084	p=.071	p=.163
1c8+NN6	p=.586		p=.104		p=.021	p=.022	p=.636						
1c16+NN6	p=.586		p<.001		p=.38	p=.643	p=.467						
2c12+NN6	p=.119		p=.009		p=.38	p=.281	p=.636						
3c12+NN6	p=.061		p=.004		p=.212	p=.643	p=.928						
Blocklists vs minimum-strength policies													
1c8+NN6												p=.074	p=.156
1c8+Xato-cifs	p=.007	p=.065	p=.734	N/A	p=.609	p=.66	p>.999	p>.999	p=.45	p=.585	p=.279		
1c8+Pwned-fs	p>.999	p=.747	p=.747	N/A	p=.609	p=.66	p=.724	p>.999	p=.654	p=.678	p=.683		
1c8+Xato-strip-cifs	p>.999	p>.999	p=.282	N/A	p=.054	p=.47	p>.999	p>.999	p=.654	p=.203	p=.658		
1c8+Xato-ciss	p=.005	p=.747	p<.001	N/A	p<.001	p<.001	p=.003	p=.01	p<.001	p=.678	p=.683		
Policies enforcing composition requirements only													
3c8	p=.558	p=.639	p=.635		p=.891	p=.124		p=.157	p=.174	p=.414	p=.102	p=.711	p=.103
4c8				N/A			p=.604						
1c8				N/A			p=.037						
***3c8+NN6* vs *1c8+NN12* (post-hoc comparison)**													
3c8+NN6													
1c8+NN12	p=.001	p=.343	p<.001	p<.001	p=.036	p=.464	p=.184	p=.821	p=.37	p=.426	p=.076	p=.125	p=.052

Table 6: Experiment 1 policy comparisons

Condition	Peto-Peto	percent @ 10^{6}	percent @ 10^{14}	Noticed rejection	Creation time	Creation difficult	Creation annoying	Stored password	Part 1 dropout	Part 2 dropout	Recall time	Recall success	Remember difficult
Blocklists vs. minimum-strength policies													
1c8+Pwned-fs		p=.89					p=.223	p=.987	p=.062	p=.689	p=.629	p=.257	p=.7
1c8+NN8	p<.001		p=.427	N/A	p=.648	p=.136							
1c8+NN10	p<.001		p=.002	N/A	p<.001	p=.027							
1c8+Xato-strip-cifs		p=.107				p=.409	p=.551	p>.999	p=.247	p=.937	p=.3	p=.261	p=.25
1c8+NN8	p=.009		p>.999	N/A	p=.122								
1c8+NN10	p<.001		p=.034	N/A	p=.066								
Minimum-strength and minimum-length requirement interactions													
1c10+NN8		p=.205		p=.318	p=.197	p=.827	p=.26	p=.865	p=.68	p=.913	p=.319	p=.644	p=.756
1c8+NN8	p=.628		p=.02										
1c10+NN10	p=.002		p=.08										
1c12+NN10	p=.925	p=.356			p=.1	p=.234	p=.66	p=.213	p=.67	p=.41	p=.285	p=.349	p=.982
1c10+NN10			p=.086	p=.009									
1c8+NN10			p<.001	p<.001									
1c8+NN10		p=.229		p=.076		p=.659	p=.323	p>.999	p=.38	p=.963	p=.46	p=.262	p=.829
1c10+NN10	p=.752		p=.048		p=.045								
1c8+NN8	p<.001		p=.032		p=.001								
Blocklist and composition requirement interactions													
1c8+Xato-strip-cifs													
4c8+Xato-strip-cifs	p=.855	p=.582	p=.66	N/A	p=.301	p=.452	p=.064	p=.758	p=.094	p=.75	p=.131	p=.459	p=.054
1c8+Pwned-fs													
4c8+Pwned-fs	p=.039	p=.027	p=.042	N/A	p=.206	p=.268	p=.94	p=.069	p=.93	p=.57	p=.775	p=.505	p=.699
1c8+Pwned-fs													
1c8+Xato-strip-cifs	p=.067	p=.141	p=.431	N/A	p=.04	p=.223	p=.474	p=.1	p=.015	p=.697	p=.263	p=.786	p=.061
4c8+Pwned-fs													
4c8+Xato-strip-cifs	p=.003	p=.135	p=.018	N/A	p=.062	p=.436	p=.018	p=.564	p=.408	p=.62	p=.898	p=.085	p=.789

Table 7: Experiment 2 policy comparisons

F.2 Details of statistical tests

Tables 8-11 contain the detailed results of each statistical test. For Fisher's exact tests, the odds ratio estimate (OR) and its 95% confidence interval are listed. For Mann-Whitney U tests, the μ location parameter estimate and its 95% confidence interval are listed.

Comparison	Family	Details	Adj. p
Tests on guessability curves (Peto-Peto)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	χˆ2=0, df=1	0.842
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	χˆ2=1.8, df=1	0.54
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	χˆ2=1.8, df=1	0.54
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	χˆ2=1.2, df=1	0.54
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	χˆ2=12, df=1	0.002
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	χˆ2=6.9, df=1	0.017
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	χˆ2=8.2, df=1	0.013
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	χˆ2=0.1, df=1	0.8
3c8+NN6 v. 1c8+NN6	Comp. reqs. for min-str.	χˆ2=1.1, df=1	0.586
3c8+NN6 v. 1c16+NN6	Comp. reqs. for min-str.	χˆ2=0.9, df=1	0.586
3c8+NN6 v. 2c12+NN6	Comp. reqs. for min-str.	χˆ2=4.2, df=1	0.119
3c8+NN6 v. 3c12+NN6	Comp. reqs. for min-str.	χˆ2=5.9, df=1	0.061
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	χˆ2=9.2, df=1	0.007
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	χˆ2=0, df=1	>.999
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	χˆ2=10.3, df=1	0.005
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	χˆ2=0, df=1	>.999
3c8 (PGS3 NN) v. 3c8 (PGS++ NN)	PGS3 NN v. PGS++ NN	χˆ2=11.9, df=1	0.001
4c8 (PGS3 NN) v. 4c8 (PGS++ NN)	PGS3 NN v. PGS++ NN	χˆ2=18.2, df=1	<.001
1c8 (PGS3 NN) v. 1c8 (PGS++ NN)	PGS3 NN v. PGS++ NN	χˆ2=0.3, df=1	0.56
Omnibus	Comp. reqs. only	χˆ2=1.2, df=2	0.558
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	χˆ2=10.8, df=1	0.001
Experiment 2			
1c8+Pwned-fs v. 1c8+NN8	Bl. v. min-str. (Pwned)	χˆ2=24.1, df=1	<.001
1c8+Pwned-fs v. 1c8+NN10	Bl. v. min-str. (Pwned)	χˆ2=64.6, df=1	<.001
1c8+Xato-scifs v. 1c8+NN8	Bl. v. min-str. (Xato)	χˆ2=6.8, df=1	0.009
1c8+Xato-scifs v. 1c8+NN10	Bl. v. min-str. (Xato)	χˆ2=40.7, df=1	<.001
1c10+NN8 v. 1c8+NN8	Min-str./len. req. ia. (1c10)	χˆ2=0.2, df=1	0.628
1c10+NN8 v. 1c10+NN10	Min-str./len. req. ia. (1c10)	χˆ2=11.1, df=1	0.002
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=0.2, df=2	0.925
1c8+NN10 v. 1c10+NN10	Min-str./len. req. ias. (1c8)	χˆ2=0.1, df=1	0.752
1c8+NN8 v. 1c8+NN10	Min-str./len. req. ias. (1c8)	χˆ2=24.8, df=1	<.001
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	χˆ2=0, df=1	0.855
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	χˆ2=4.3, df=1	0.039
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	χˆ2=3.4, df=1	0.067
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	χˆ2=9.1, df=1	0.003
Tests on the proportions of password guessed at 10ˆ6 guess cutoff (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.18 ([0.47, 2.92])	0.835
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=0.71 ([0.31, 1.57])	0.46
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=0 ([0.00, 0.22])	<.001
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=0.16 ([0.03, 0.54])	0.001
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0 ([0.00, 0.25])	<.001
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=4.51 ([1.20, 25.17])	0.048
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=Inf ([0.35, Inf])	0.498
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=0 ([0.00, 2.53])	0.498
Omnibus	Comp. reqs. for min-str.	χˆ2=5.7, df=4	0.224
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=4.48 ([1.19, 25.00])	0.065
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.99 ([0.13, 7.48])	>.999
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0 ([0.00, 2.83])	0.747
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=0 ([0.00, 2.51])	0.747
Omnibus	Comp. reqs. only	χˆ2=0.9, df=2	0.639
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=5.6, df=5	0.343
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=0.2, df=2	0.89
Omnibus	Bl. v. min-str. (Xato)	χˆ2=4.5, df=2	0.107
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=3.2, df=2	0.205
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=2.1, df=2	0.356
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=3, df=2	0.229
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=1.49 ([0.42, 5.84])	0.582
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.24 ([0.04, 0.91])	0.027
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=0.37 ([0.06, 1.54])	0.141
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=2.29 ([0.73, 8.50])	0.135
Tests on the proportions of participants that noticed bl./min-str. rejection (Chi-square, FET)			
Experiment 1			
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=0.48 ([0.21, 1.05])	0.051
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.28 ([0.15, 0.50])	<.001
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=0.31 ([0.16, 0.59])	<.001
Omnibus	Comp. reqs. for min-str.	χˆ2=3.8, df=4	0.436
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	OR=0.33 ([0.21, 0.50])	<.001
Experiment 2			
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=2.3, df=2	0.318
1c12+NN10 v. 1c10+NN10	Min-str./len. req. ias. (1c12)	OR=1.54 ([1.10, 2.16])	0.009
1c12+NN10 v. 1c8+NN10	Min-str./len. req. ias. (1c12)	OR=1.88 ([1.35, 2.64])	<.001
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=5.2, df=2	0.076

Table 8: Detailed statistical test results (1)

Comparison	Family	Details	Adj. p
Tests on the proportions of password guessed at 10ˆ14 guess cutoff (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=0.77 ([0.54, 1.08])	0.114
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=0.94 ([0.67, 1.32])	0.737
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=0.78 ([0.55, 1.09])	0.267
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=0.75 ([0.53, 1.05])	0.25
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0.35 ([0.24, 0.51])	<.001
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=1.25 ([0.88, 1.78])	0.399
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=2.15 ([1.46, 3.20])	<.001
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=1.04 ([0.73, 1.47])	0.865
3c8+NN6 v. 1c8+NN6	Comp. reqs. for min-str.	OR=0.75 ([0.53, 1.07])	0.104
3c8+NN6 v. 1c16+NN6	Comp. reqs. for min-str.	OR=3.91 ([2.56, 6.05])	<.001
3c8+NN6 v. 2c12+NN6	Comp. reqs. for min-str.	OR=1.67 ([1.16, 2.41])	0.009
3c8+NN6 v. 3c12+NN6	Comp. reqs. for min-str.	OR=1.84 ([1.26, 2.71])	0.004
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=0.94 ([0.66, 1.33])	0.734
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.75 ([0.53, 1.05])	0.282
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.35 ([0.23, 0.51])	<.001
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=0.77 ([0.55, 1.09])	0.282
Omnibus	Comp. reqs. only	χˆ2=0.9, df=2	0.635
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	OR=4.68 ([2.98, 7.49])	<.001
Experiment 2			
1c8+Pwned-fs v. 1c8+NN8	Bl. v. min-str. (Pwned)	OR=1.12 ([0.84, 1.51])	0.427
1c8+Pwned-fs v. 1c8+NN10	Bl. v. min-str. (Pwned)	OR=1.62 ([1.2, 2.2])	0.002
1c8+Xato-scifs v. 1c8+NN8	Bl. v. min-str. (Xato)	OR=1 ([0.74, 1.34])	>.999
1c8+Xato-scifs v. 1c8+NN10	Bl. v. min-str. (Xato)	OR=1.44 ([1.06, 1.96])	0.034
1c10+NN8 v. 1c8+NN8	Min-str./len. req. ia. (1c10)	OR=1.49 ([1.10, 2.04])	0.02
1c10+NN8 v. 1c10+NN10	Min-str./len. req. ia. (1c10)	OR=1.33 ([0.96, 1.85])	0.08
1c12+NN10 v. 1c10+NN10	Min-str./len. req. ias. (1c12)	OR=1.36 ([0.96, 1.94])	0.086
1c12+NN10 v. 1c8+NN10	Min-str./len. req. ias. (1c12)	OR=1.87 ([1.33, 2.65])	<.001
1c8+NN10 v. 1c10+NN10	Min-str./len. req. ias. (1c8)	OR=1.38 ([1.00, 1.91])	0.048
1c8+NN8 v. 1c8+NN10	Min-str./len. req. ias. (1c8)	OR=1.45 ([1.06, 1.97])	0.032
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.93 ([0.69, 1.26])	0.66
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.74 ([0.55, 0.99])	0.042
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=1.12 ([0.84, 1.50])	0.431
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=1.43 ([1.05, 1.93])	0.018
Tests on password-creation time (KW, MWU)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	W=48150, μ=0.6 ([-7.8, 9.2])	0.895
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	W=50088, μ=9.2 ([0.5, 18.2])	0.114
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	W=47202, μ=1.5 ([-7.4, 10.8])	0.722
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	W=46983, μ=-5.1 ([-15.1, 4.3])	0.558
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	W=28044, μ=-46.3 ([-61.5, -32.1])	<.001
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	W=37932, μ=-15.2 ([-25.1, -5.7])	0.003
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	W=29159, μ=-41.0 ([-56.4, -26.9])	<.001
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	W=42266, μ=-7.4 ([-17.0, 2.2])	0.142
3c8+NN6 v. 1c8+NN6	Comp. reqs. for min-str.	W=43247, μ=13.2 ([3.9, 22.5])	0.021
3c8+NN6 v. 1c16+NN6	Comp. reqs. for min-str.	W=34605, μ=-4.9 ([-14.3, 4.5])	0.38
3c8+NN6 v. 2c12+NN6	Comp. reqs. for min-str.	W=41301, μ=6.2 ([-3.1, 15.6])	0.38
3c8+NN6 v. 3c12+NN6	Comp. reqs. for min-str.	W=30818, μ=-9.2 ([-19.5, 0.8])	0.212
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	W=43028, μ=3.4 ([-4.5, 11.6])	0.609
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	W=39777, μ=-11.6 ([-21.7, -2.0])	0.054
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	W=23233, μ=-52.5 ([-67.8, -38.6])	<.001
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	W=39974, μ=-4.6 ([-13.5, 4.2])	0.609
Omnibus	Comp. reqs. only	χˆ2=0.231, df=2	0.891
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	W=30801, μ=-11.6 ([-23.1, -0.8])	0.036
Experiment 2			
1c8+Pwned-fs v. 1c8+NN8	Bl. v. min-str. (Pwned)	W=81334, μ=-1.8 ([-9.4, 5.8])	0.648
1c8+Pwned-fs v. 1c8+NN10	Bl. v. min-str. (Pwned)	W=69655, μ=-18.4 ([-27.7, -9.6])	<.001
1c8+Xato-scifs v. 1c8+NN8	Bl. v. min-str. (Xato)	W=87868, μ=6.5 ([-1.8, 15.2])	0.122
1c8+Xato-scifs v. 1c8+NN10	Bl. v. min-str. (Xato)	W=76348, μ=-10.1 ([-19.5, -0.8])	0.066
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=3.25, df=2	0.197
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=4.61, df=2	0.1
1c8+NN10 v. 1c10+NN10	Min-str./len. req. ias. (1c8)	W=83571, μ=9.4 ([0.2, 18.8])	0.045
1c8+NN8 v. 1c8+NN10	Min-str./len. req. ias. (1c8)	W=62294, μ=-16.6 ([-26.4, -7.5])	0.001
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	W=83838, μ=-4.7 ([-13.7, 4.2])	0.301
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	W=77992, μ=-5.0 ([-12.8, 2.9])	0.206
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	W=86778, μ=-8.4 ([-16.7, -0.4])	0.04
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	W=70276, μ=-8.3 ([-17.4, 0.4])	0.062
Tests on password-creation difficulty (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.02 ([0.7, 1.5])	0.926
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=0.94 ([0.64, 1.38])	>.999
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=1.01 ([0.69, 1.49])	>.999
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=0.86 ([0.59, 1.25])	>.999
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0.34 ([0.23, 0.49])	<.001
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=0.92 ([0.63, 1.35])	0.808
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.39 ([0.27, 0.56])	<.001
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=0.85 ([0.58, 1.25])	0.808
3c8+NN6 v. 1c8+NN6	Comp. reqs. for min-str.	OR=1.72 ([1.16, 2.56])	0.022
3c8+NN6 v. 1c16+NN6	Comp. reqs. for min-str.	OR=0.83 ([0.57, 1.19])	0.643
3c8+NN6 v. 2c12+NN6	Comp. reqs. for min-str.	OR=1.38 ([0.95, 2.03])	0.281
3c8+NN6 v. 3c12+NN6	Comp. reqs. for min-str.	OR=1.11 ([0.75, 1.63])	0.643
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=0.82 ([0.55, 1.23])	0.66
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.76 ([0.51, 1.12])	0.47
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.3 ([0.20, 0.44])	<.001
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=0.89 ([0.59, 1.33])	0.66
Omnibus	Comp. reqs. only	χˆ2=4.2, df=2	0.124
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	OR=0.87 ([0.60, 1.27])	0.464
Experiment 2			
1c8+Pwned-fs v. 1c8+NN8	Bl. v. min-str. (Pwned)	OR=0.78 ([0.57, 1.08])	0.136
1c8+Pwned-fs v. 1c8+NN10	Bl. v. min-str. (Pwned)	OR=0.68 ([0.50, 0.93])	0.027
Omnibus	Bl. v. min-str. (Xato)	χˆ2=1.8, df=2	0.409
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.4, df=2	0.827
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=2.9, df=2	0.234
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=0.8, df=2	0.659
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.89 ([0.65, 1.21])	0.452
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.84 ([0.61, 1.15])	0.268
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=0.83 ([0.61, 1.13])	0.223
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=0.88 ([0.64, 1.21])	0.436

Table 9: Detailed statistical test results (2)

Comparison	Family	Details	Adj. p
Tests on password-creation annoyance (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.15 ([0.82, 1.62])	0.449
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=0.82 ([0.57, 1.19])	0.574
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=1 ([0.69, 1.44])	>.999
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=0.73 ([0.51, 1.04])	0.215
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0.44 ([0.31, 0.63])	<.001
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=0.89 ([0.62, 1.26])	0.544
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.61 ([0.43, 0.86])	0.011
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=0.73 ([0.51, 1.05])	0.161
3c8+NN6 v. 1c8+NN6	Comp. reqs. for min-str.	OR=1.26 ([0.87, 1.81])	0.636
3c8+NN6 v. 1c16+NN6	Comp. reqs. for min-str.	OR=0.76 ([0.53, 1.09])	0.467
3c8+NN6 v. 2c12+NN6	Comp. reqs. for min-str.	OR=1.24 ([0.86, 1.78])	0.636
3c8+NN6 v. 3c12+NN6	Comp. reqs. for min-str.	OR=1.02 ([0.71, 1.48])	0.928
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=1.03 ([0.71, 1.48])	>.999
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.91 ([0.64, 1.30])	>.999
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.55 ([0.38, 0.79])	0.003
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=1.24 ([0.86, 1.80])	0.724
3c8 v. 4c8	Comp. reqs. only	OR=1.1 ([0.78, 1.57])	0.604
3c8 v. 1c8	Comp. reqs. only	OR=1.53 ([1.07, 2.19])	0.037
3c8NN6 vs 1c8NN12	Comp. reqs. for min-str. (post-hoc)	OR=0.79 ([0.55, 1.13])	0.184
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=3, df=2	0.223
Omnibus	Bl. v. min-str. (Xato)	χˆ2=1.2, df=2	0.551
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=2.7, df=2	0.26
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=0.8, df=2	0.66
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=2.3, df=2	0.323
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.76 ([0.57, 1.02])	0.064
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.98 ([0.72, 1.33])	0.94
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=0.9 ([0.67, 1.20])	0.474
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=0.7 ([0.52, 0.94])	0.018
Tests on password storage after creation (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.15 ([0.79, 1.66])	0.941
Omnibus	Impact of bl. (1c8)	χˆ2=10.4, df=4	0.07
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=0.97 ([0.67, 1.41])	>.999
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=1.05 ([0.72, 1.52])	>.999
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=1 ([0.70, 1.43])	>.999
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0.6 ([0.4, 0.9])	0.038
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=1.03 ([0.72, 1.48])	>.999
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.6 ([0.41, 0.89])	0.031
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=0.95 ([0.66, 1.37])	>.999
Omnibus	Comp. reqs. for min-str.	χˆ2=2.2, df=4	0.705
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=0.88 ([0.60, 1.28])	>.999
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.9 ([0.62, 1.30])	>.999
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.54 ([0.36, 0.82])	0.01
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=0.94 ([0.65, 1.38])	>.999
Omnibus	Comp. reqs. only	χˆ2=5.1, df=2	0.157
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=3.6, df=5	0.821
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=1.4, df=2	0.987
Omnibus	Bl. v. min-str. (Xato)	χˆ2=1.4, df=2	>.999
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.3, df=2	0.865
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=4.5, df=2	0.213
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=0.4, df=2	>.999
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=1.15 ([0.83, 1.60])	0.758
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.7 ([0.50, 0.98])	0.069
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=0.73 ([0.53, 1.01])	0.1
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=1.2 ([0.86, 1.69])	0.564
Tests on Part 1 participant dropout rate (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.52 ([0.95, 2.45])	0.07
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=1.19 ([0.74, 1.91])	0.988
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.38 ([0.25, 0.58])	<.001
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=1.06 ([0.66, 1.73])	0.988
Omnibus	Comp. reqs. for min-str.	χˆ2=5.1, df=4	0.272
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=0.69 ([0.41, 1.14])	0.45
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=0.82 ([0.49, 1.36])	0.654
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.31 ([0.20, 0.49])	<.001
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=0.77 ([0.46, 1.28])	0.654
Omnibus	Comp. reqs. only	χˆ2=3.5, df=2	0.174
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=5.4, df=5	0.37
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=5.6, df=2	0.062
Omnibus	Bl. v. min-str. (Xato)	χˆ2=2.8, df=2	0.247
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.8, df=2	0.68
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=0.8, df=2	0.67
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=1.9, df=2	0.38
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.71 ([0.47, 1.07])	0.094
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.97 ([0.68, 1.40])	0.93
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=1.59 ([1.08, 2.36])	0.015
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=1.17 ([0.8, 1.7])	0.408

Table 10: Detailed statistical test results (3)

Comparison	Family	Details	Adj. p
Tests on Part 2 participant dropout rate (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=0.66 ([0.43, 1.00])	0.047
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	OR=1.42 ([0.92, 2.21])	0.290
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	OR=1.32 ([0.86, 2.02])	0.436
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	OR=1.65 ([1.07, 2.57])	0.086
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	OR=0.89 ([0.59, 1.35])	0.616
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=1.16 ([0.72, 1.86])	0.738
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=0.54 ([0.34, 0.85])	0.015
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=1.25 ([0.79, 2.00])	0.738
Omnibus	Comp. reqs. for min-str.	χˆ2=4.2, df=4	0.385
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	OR=1.35 ([0.86, 2.12])	0.585
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	OR=1.56 ([1.00, 2.46])	0.203
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	OR=0.84 ([0.55, 1.29])	0.678
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	OR=1.25 ([0.80, 1.94])	0.678
Omnibus	Comp. reqs. only	χˆ2=1.8, df=2	0.414
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=4.9, df=5	0.426
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=0.7, df=2	0.689
Omnibus	Bl. v. min-str. (Xato)	χˆ2=0.1, df=2	0.937
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.2, df=2	0.913
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=1.8, df=2	0.41
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=0.1, df=2	0.963
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.94 ([0.68, 1.30])	0.75
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=1.11 ([0.80, 1.54])	0.570
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=1.07 ([0.78, 1.47])	0.697
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=0.91 ([0.65, 1.28])	0.62
Tests on password-recall time in Part 2 (KW, MWU)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	W=4512, μ=3.3 ([-0.8, 7.3])	0.101
1c8 v. 1c8+Xato-cifs	Impact of bl. (1c8)	W=3703, μ=1.8 ([-1.9, 5.5])	0.570
1c8 v. 1c8+Pwned-fs	Impact of bl. (1c8)	W=3655, μ=-2.8 ([-6.8, 0.9])	0.418
1c8 v. 1c8+Xato-scifs	Impact of bl. (1c8)	W=3541, μ=-4.0 ([-8.5, 0.3])	0.290
1c8 v. 1c8+Xato-ciss	Impact of bl. (1c8)	W=1629, μ=-2.8 ([-8.3, 2.2])	0.570
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	W=2649, μ=-5.7 ([-10.4, -1.6])	0.016
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	W=1957, μ=0.8 ([-4.6, 6.5])	>.999
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	W=4194, μ=-0.7 ([-5.3, 3.3])	>.999
Omnibus	Comp. reqs. for min-str.	χˆ2=8.23, df=4	0.084
1c8+NN6 v. 1c8+Xato-cifs	Bl. v. min-str.	W=3809, μ=3.0 ([-0.2, 6.4])	0.279
1c8+NN6 v. 1c8+Xato-scifs	Bl. v. min-str.	W=3573, μ=-2.6 ([-7.1, 1.4])	0.658
1c8+NN6 v. 1c8+Xato-ciss	Bl. v. min-str.	W=1629, μ=-1.6 ([-6.6, 3.1])	0.683
1c8+NN6 v. 1c8+Pwned-fs	Bl. v. min-str.	W=3669, μ=-1.7 ([-5.4, 2.0])	0.683
Omnibus	Comp. reqs. only	χˆ2=4.57, df=2	0.102
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=9.98, df=5	0.076
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=0.927, df=2	0.629
Omnibus	Bl. v. min-str. (Xato)	χˆ2=2.41, df=2	0.3
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=2.28, df=2	0.319
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=2.51, df=2	0.285
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=1.55, df=2	0.46
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	W=4798, μ=-3.0 ([-7.1, 0.9])	0.131
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	W=5188, μ=-0.6 ([-5.0, 4.1])	0.775
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	W=6514, μ=2.4 ([-1.5, 6.9])	0.263
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	W=4790, μ=-0.3 ([-4.7, 4.2])	0.898
Tests on recall success in Part 2 (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=1.28 ([0.65, 2.54])	0.524
Omnibus	Impact of bl. (1c8)	χˆ2=8.5, df=4	0.076
1c8+Xato-scifs v. 1c8+Xato-cifs	Bl. reqs. for 1c8	OR=0.89 ([0.44, 1.78])	0.745
1c8+Xato-scifs v. 1c8+Xato-ciss	Bl. reqs. for 1c8	OR=1.81 ([0.85, 3.83])	0.31
1c8+Xato-scifs v. 1c8+Pwned-fs	Bl. reqs. for 1c8	OR=1.66 ([0.79, 3.63])	0.332
Omnibus	Comp. reqs. for min-str.	χˆ2=8.6, df=4	0.071
Omnibus	Bl. v. min-str.	χˆ2=8.5, df=4	0.074
Omnibus	Comp. reqs. only	χˆ2=0.7, df=2	0.711
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=8.6, df=5	0.125
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=2.7, df=2	0.257
Omnibus	Bl. v. min-str. (Xato)	χˆ2=2.7, df=2	0.261
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.9, df=2	0.644
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=2.1, df=2	0.349
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=2.7, df=2	0.262
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.78 ([0.41, 1.46])	0.459
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=1.2 ([0.69, 2.11])	0.505
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=0.93 ([0.52, 1.64])	0.786
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=0.6 ([0.32, 1.11])	0.085
Tests on password-remembrance difficulty (Chi-square, FET)			
Experiment 1			
3c8 v. 3c8+Xato-cifs	Impact of bl. (3c8)	OR=0.65 ([0.34, 1.25])	0.176
Omnibus	Impact of bl. (1c8)	χˆ2=6.3, df=4	0.179
Omnibus	Bl. reqs. for 1c8	χˆ2=5.5, df=2	0.064
Omnibus	Comp. reqs. for min-str.	χˆ2=6.5, df=4	0.163
Omnibus	Bl. v. min-str.	χˆ2=6.6, df=4	0.156
Omnibus	Comp. reqs. only	χˆ2=4.5, df=2	0.103
Omnibus	Comp. reqs. for min-str. (post-hoc)	χˆ2=10.9, df=5	0.052
Experiment 2			
Omnibus	Bl. v. min-str. (Pwned)	χˆ2=0.7, df=2	0.7
Omnibus	Bl. v. min-str. (Xato)	χˆ2=2.8, df=2	0.25
Omnibus	Min-str./len. req. ia. (1c10)	χˆ2=0.6, df=2	0.756
Omnibus	Min-str./len. req. ias. (1c12)	χˆ2=0, df=2	0.982
Omnibus	Min-str./len. req. ias. (1c8)	χˆ2=0.4, df=2	0.829
1c8+Xato-scifs v. 4c8+Xato-scifs	Bl./comp. req. ias. (Xato)	OR=0.580 ([0.32, 1.03])	0.054
1c8+Pwned-fs v. 4c8+Pwned-fs	Bl./comp. req. ias. (Pwned)	OR=0.9 ([0.53, 1.55])	0.699
1c8+Pwned-fs v. 1c8+Xato-scifs	Bl./comp. req. ias. (1c8)	OR=1.72 ([0.98, 3.05])	0.061
4c8+Pwned-fs v. 4c8+Xato-scifs	Bl./comp. req. ias. (4c8)	OR=1.1 ([0.63, 1.92])	0.789

Table 11: Detailed statistical test results (4)

REFERENCES

[1] andr0id. 2004. Word lists. http://www.outpost9.com/files/WordLists.html.

[2] bbondy. 2015. bloom-filter-js. https://github.com/bbondy/bloom-filter-js.

[3] Mark Burnett. 2015. Today I am releasing ten million passwords. https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495.

[4] Xavier De Carné De Carnavalet and Mohammad Mannan. 2014. From Very Weak to Very Strong: Analyzing Password-Strength Meters. In *NDSS*. 23–26.

[5] Matteo Dell'Amico and Maurizio Filippone. 2015. Monte Carlo strength evaluation: Fast and reliable password checking. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 158–169.

[6] Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. 2013. On the Ecological Validity of a Password Study. In *Proceedings of the Ninth Symposium on Usable Privacy and Security* (Newcastle, United Kingdom) *(SOUPS '13)*. ACM, New York, NY, USA, Article 13, 13 pages. https://doi.org/10.1145/2501604.2501617

[7] Dinei Florêncio, Cormac Herley, and Paul C van Oorschot. 2014. An Administrator's Guide to Internet Password Research. In *28th Large Installation System Administration Conference (LISA14)*. USENIX Association, Seattle, WA, 44–61.

[8] Hana Habib, Jessica Colnago, William Melicher, Blase Ur, Sean Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Cranor. 2017. Password creation in the presence of blacklists. In *Proceedings of Usable Security (USEC) 2017*. Internet Society. https://doi.org/10.14722/usec.2017.23043

[9] Troy Hunt. 2018. Enhancing Pwned Passwords Privacy by Exclusively Supporting Anonymity. https://www.troyhunt.com/enhancing-pwned-passwords-privacy-by-exclusively-supporting-anonymity.

[10] Troy Hunt. 2019. Pwned Passwords API. https://haveibeenpwned.com/Passwords.

[11] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *2012 IEEE Symposium on Security and Privacy*. 523–537. https://doi.org/10.1109/SP.2012.38

[12] Saranga Komanduri, Richard Shay, Lorrie Faith Cranor, Cormac Herley, and Stuart Schechter. 2014. Telepathwords: Preventing Weak Passwords by Reading Users' Minds. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 591–606.

[13] 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 *CHI '11: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2595–2604.

[14] 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 *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (Berlin, Germany) *(CCS '13)*. ACM, New York, NY, USA, 173–186. https://doi.org/10.1145/2508859.2516726

[15] 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 *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. ACM, New York, NY, USA, 527–539. https://doi.org/10.1145/2858036.2858384

[16] William Melicher, Blase Ur, Sean M Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, lean, and accurate: Modeling password guessability using neural networks. In *Proceedings of the 25th USENIX Security Symposium*.

[17] Randall Munroe. 2011. Password strength. https://xkcd.com/936/.

[18] National Institute of Standards and Technology (NIST). 2004. SP 800-63 Ver. 1.0: Electronic Authentication Guideline. https://csrc.nist.gov/publications/detail/sp/800-63/ver-10/archive/2004-06-30.

[19] National Institute of Standards and Technology (NIST). 2017. SP 800-63B: Digital Identity Guidelines: Authentication and Lifecycle Management. https://doi.org/10.6028/NIST.SP.800-63-3. Updated Dec 2017.

[20] Openwall. 2003. Openwall file archive. http://download.openwall.net/pub/wordlists/languages/English/4-extra/lower.gz.

[21] Password Research Team at Carnegie Mellon University. 2019. Password Guessability Service. https://pgs.ece.cmu.edu.

[22] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. 2017. Let's go in for a closer look: Observing passwords in their natural habitat. In *CCS*.

[23] Robert W Proctor, Mei-Ching Lien, Kim-Phuong L Vu, E Eugene Schultz, and Gavriel Salvendy. 2002. Improving computer security for authentication of users: Influence of proactive password restrictions. *Behavior Research Methods, Instruments, & Computers* 34, 2 (2002), 163–169.

[24] Sean M Segreti, William Melicher, Saranga Komanduri, Darya Melicher, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L Mazurek. 2017. Diversify to survive: Making passwords stronger with adaptive policies. In *SOUPS '17: Proceedings of the 13th Symposium on Usable Privacy and Security*. USENIX.

[25] 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 *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2903–2912.

[26] 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 *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 2927–2936. https://doi.org/10.1145/2556288.2557377

[27] 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. https://doi.org/10.1145/2891411

[28] Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Harold Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, and William Melicher. 2017. Design and evaluation of a data-driven password meter. In *CHI'17: 35th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3775–3786.

[29] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L 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 *Proceedings of the 21st USENIX Security Symposium*. USENIX Association.

[30] Blase Ur, Sean M Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L Mazurek, William Melicher, and Richard Shay. 2015. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In *Proceedings of the 24th USENIX Security Symposium*. USENIX.

[31] Kim-Phuong L Vu, Robert W Proctor, Abhilasha Bhargav-Spantzel, Bik-Lam Belin Tai, Joshua Cook, and E Eugene Schultz. 2007. Improving password security and memorability to protect personal and organizational information. *International Journal of Human-Computer Studies* 65, 8 (2007), 744–757.

[32] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. 2016. Understanding password choices: How frequently entered passwords are re-used across websites. In *Twelfth Symposium on Usable Privacy and Security SOUPS*.

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

[34] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 157–173.