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:

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 implementations 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

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CCS ’20, November 9–13, 2020, Virtual Event, USA
© 2020 Copyright held by the owner/author(s).
https://doi.org/10.1145/3372297.3417882
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 full-string-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 “Password!” 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 blocked 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,
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, block-list, 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

and wordlists based on the OpenWall dictionary, which was found to be relatively ineffective [12, 13].
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 10⁶ guesses.

We tested four minimum-strength thresholds, ranging from 10⁶ to 10¹² 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.

#### 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 10⁶ 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

---

¹This 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].

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

³We 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 Powed 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).

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). 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

\[\text{As we employ real-time feedback in our password meter, our study data do not include the notion of a password-creation attempt. However, this concept is closely related to creation time and creation annoyance/difficulty.}\]
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. 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. These thresholds have been used in prior work as estimates of how many guesses an online and an offline attacker could make. 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. These thresholds have been used in prior work as estimates of how many guesses an online and an offline attacker could make. 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.

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

---

1 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.
2 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 re-
quirements to provide adequate protection against online guessing
attacks. In Experiment 2 we explore in more depth 1c8 minimum-
length policies that provide strong protection against both online
and offline guessing attacks. We also extensively analyze interac-
tion 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 demon-
strated 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 compar-
isons were made with respect to Xato-strip-cifs, which was rec-
ommended in prior work [8]. We found that case-sensitive, full-
string matching against very large blocklists of leaked pass-
words 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 on-
line 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

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 algo-
rum 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. Par-
ticipants 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 at-
tacks 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

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

### 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.

<table>
<thead>
<tr>
<th>Condition</th>
<th># in Part 1</th>
<th># in Part 2</th>
<th>Part 1 dropout</th>
<th>Creation time</th>
<th>Creation difficult</th>
<th>Creation annoying</th>
<th>Guessed @ 10⁶</th>
<th>Guessed @ 10¹⁴</th>
<th>Noticed reject</th>
<th>Stored pwd</th>
<th>Recall success</th>
<th>Recall time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experiment 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMU</td>
<td>290</td>
<td>228</td>
<td>14%</td>
<td>104 s</td>
<td>34%</td>
<td>42%</td>
<td>1.2%</td>
<td>36.3%</td>
<td>50%</td>
<td>57%</td>
<td>74%</td>
<td>26 s</td>
</tr>
<tr>
<td>1c8</td>
<td>284</td>
<td>235</td>
<td>15%</td>
<td>78 s</td>
<td>25%</td>
<td>37%</td>
<td>4.5%</td>
<td>44.8%</td>
<td>-</td>
<td>52%</td>
<td>80%</td>
<td>24 s</td>
</tr>
<tr>
<td>4c8</td>
<td>297</td>
<td>237</td>
<td>10%</td>
<td>84 s</td>
<td>31%</td>
<td>35%</td>
<td>5.4%</td>
<td>48.4%</td>
<td>-</td>
<td>44%</td>
<td>76%</td>
<td>27 s</td>
</tr>
<tr>
<td>1c8+NN6</td>
<td>318</td>
<td>250</td>
<td>13%</td>
<td>86 s</td>
<td>25%</td>
<td>28%</td>
<td>6.3%</td>
<td>48.2%</td>
<td>-</td>
<td>53%</td>
<td>78%</td>
<td>21 s</td>
</tr>
<tr>
<td>3c8+NN6</td>
<td>264</td>
<td>213</td>
<td>15%</td>
<td>92 s</td>
<td>33%</td>
<td>38%</td>
<td>0.4%</td>
<td>41.3%</td>
<td>22%</td>
<td>56%</td>
<td>79%</td>
<td>21 s</td>
</tr>
<tr>
<td>1c8+NN12</td>
<td>261</td>
<td>213</td>
<td>14%</td>
<td>100 s</td>
<td>36%</td>
<td>44%</td>
<td>0.4%</td>
<td>13%</td>
<td>46%</td>
<td>56%</td>
<td>75%</td>
<td>28 s</td>
</tr>
<tr>
<td>1c8+NN6</td>
<td>288</td>
<td>229</td>
<td>10%</td>
<td>73 s</td>
<td>22%</td>
<td>33%</td>
<td>1%</td>
<td>48.3%</td>
<td>20%</td>
<td>50%</td>
<td>79%</td>
<td>21 s</td>
</tr>
<tr>
<td>3c12+NN6</td>
<td>257</td>
<td>212</td>
<td>15%</td>
<td>99 s</td>
<td>31%</td>
<td>37%</td>
<td>0%</td>
<td>27.6%</td>
<td>16%</td>
<td>53%</td>
<td>75%</td>
<td>25 s</td>
</tr>
<tr>
<td>1c16+NN6</td>
<td>276</td>
<td>209</td>
<td>11%</td>
<td>97 s</td>
<td>37%</td>
<td>45%</td>
<td>0.4%</td>
<td>15.2%</td>
<td>16%</td>
<td>50%</td>
<td>82%</td>
<td>25 s</td>
</tr>
<tr>
<td>2c12+NN6</td>
<td>294</td>
<td>231</td>
<td>13%</td>
<td>86 s</td>
<td>26%</td>
<td>33%</td>
<td>0%</td>
<td>29.6%</td>
<td>20%</td>
<td>50%</td>
<td>66%</td>
<td>23 s</td>
</tr>
<tr>
<td>3c8+Xato-cifs</td>
<td>337</td>
<td>256</td>
<td>10%</td>
<td>81 s</td>
<td>25%</td>
<td>34%</td>
<td>3.8%</td>
<td>51.4%</td>
<td>23%</td>
<td>48%</td>
<td>76%</td>
<td>20 s</td>
</tr>
<tr>
<td>1c8+Xato-cifs</td>
<td>287</td>
<td>241</td>
<td>14%</td>
<td>70 s</td>
<td>26%</td>
<td>32%</td>
<td>4.5%</td>
<td>46.6%</td>
<td>32%</td>
<td>54%</td>
<td>77%</td>
<td>19 s</td>
</tr>
<tr>
<td>1c8+Pwned-fs</td>
<td>292</td>
<td>242</td>
<td>13%</td>
<td>85 s</td>
<td>24%</td>
<td>28%</td>
<td>0%</td>
<td>41.9%</td>
<td>23%</td>
<td>52%</td>
<td>86%</td>
<td>23 s</td>
</tr>
<tr>
<td>1c8+Xato-strip-cifs</td>
<td>311</td>
<td>267</td>
<td>12%</td>
<td>93 s</td>
<td>27%</td>
<td>35%</td>
<td>1%</td>
<td>41%</td>
<td>50%</td>
<td>53%</td>
<td>79%</td>
<td>25 s</td>
</tr>
<tr>
<td>1c8+Xato-ciss</td>
<td>261</td>
<td>200</td>
<td>26%</td>
<td>139 s</td>
<td>49%</td>
<td>47%</td>
<td>0%</td>
<td>24.4%</td>
<td>78%</td>
<td>65%</td>
<td>67%</td>
<td>23 s</td>
</tr>
</tbody>
</table>

| **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 |

![Figure 3: Min-auto guess numbers for policies containing NN6 requirements and varying composition requirements.](image-url)

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.

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 1014 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 1014 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).

<table>
<thead>
<tr>
<th>Condition</th>
<th>Peto-Peto</th>
<th>percent @ 10^6</th>
<th>percent @ 10^14</th>
<th>Creation time</th>
<th>Creation difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RQ4.B</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c8+Pwned-fs</td>
<td>p&lt;.001</td>
<td>0.7%</td>
<td>43%</td>
<td>83 s</td>
<td>25%</td>
</tr>
<tr>
<td>1c8+NN8</td>
<td>p&lt;.001</td>
<td>0.8%</td>
<td>40.2%</td>
<td>86 s</td>
<td>30%</td>
</tr>
<tr>
<td>1c8+NN10</td>
<td>p&lt;.001</td>
<td>0.5%</td>
<td>31.7%</td>
<td>109 s</td>
<td>33%</td>
</tr>
<tr>
<td>1c8+Xato-strip-cifs</td>
<td></td>
<td>2%</td>
<td>40.1%</td>
<td>97 s</td>
<td>29%</td>
</tr>
<tr>
<td>1c8+NN8</td>
<td>p=.009</td>
<td>0.8%</td>
<td>40.2%</td>
<td>86 s</td>
<td>30%</td>
</tr>
<tr>
<td>1c8+NN10</td>
<td>p&lt;.001</td>
<td>0.5%</td>
<td>31.7%</td>
<td>109 s</td>
<td>33%</td>
</tr>
</tbody>
</table>

| **RQ5**            |           |                |                 |               |                    |
| 1c8+NN8            | p=.628    | 0.8%           | 40.2%           | 86 s          | 30%                |
| 1c10+NN8           | p=.002    | 0.8%           | 40.2%           | 86 s          | 30%                |
| 1c10+NN10          | p=.925    | 0.3%           | 19.8%           | 95 s          | 28%                |
| 1c12+NN10          | p=.001    | 0.5%           | 31.7%           | 109 s         | 33%                |
| 1c8+NN10           | p=.752    | 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.

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. 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 matching a set that has been previously compromised in a public leak. Although further research is needed to confirm this, our results suggest that password subjects 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
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 $NN6$ minimum-strength policy not only prevents the most predictable $10^6$ 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 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 $10^7$ 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.

<table>
<thead>
<tr>
<th>Policy</th>
<th>PGS++ (old)</th>
<th>PGS3 (new)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1c8</td>
<td>73.4 million</td>
<td>32.8 million</td>
</tr>
<tr>
<td>1c16</td>
<td>2.5 million</td>
<td>1.5 million</td>
</tr>
<tr>
<td>2c12</td>
<td>13.3 million</td>
<td>5.4 million</td>
</tr>
<tr>
<td>3c8</td>
<td>13.6 million</td>
<td>5.7 million</td>
</tr>
<tr>
<td>3c12</td>
<td>4.1 million</td>
<td>1.8 million</td>
</tr>
<tr>
<td>4c8</td>
<td>311 thousand</td>
<td>599 thousand</td>
</tr>
</tbody>
</table>

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].

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 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 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: $\chi=11.9, p=.001$; 4c8: $\chi=18.2, p<.001$). 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.

![Image 342x473 to 534x586]

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.

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.

![Image 342x473 to 534x586]

Figure 9: Password guessability against minimum-strength-aware, minimum-strength-oblivious, and maximum-strength-aware attackers. Both types of attackers order password guesses from most probable to least probable. The maximum-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.

<table>
<thead>
<tr>
<th>Without meter</th>
<th>With meter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>2012</td>
</tr>
<tr>
<td>One class</td>
<td>38%</td>
</tr>
<tr>
<td>Two classes</td>
<td>46%</td>
</tr>
<tr>
<td>Three classes</td>
<td>12%</td>
</tr>
<tr>
<td>Four classes</td>
<td>4%</td>
</tr>
</tbody>
</table>

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).

9We 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.
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

0. 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 keep this window open as you complete the study. When you are finished, return to this page to paste the completion code into the box.

[see Appendix E.2]

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 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 entry form]

4. Part 1 password-recall task

[password entry form]

5. Part 1 survey

[see Appendix E.3]

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 fund

[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][applet]

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 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.

Because of the colored bar, I created a different password than I would have otherwise.

[Strongly disagree to Strongly agree] (5-point scale)

Do you have any other thoughts about the colored bar?

[Text box][applet]

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

The colored bar helps 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.

Because of the colored bar, I created a different password than I would have otherwise.

[Strongly disagree to Strongly agree] (5-point scale)

Do you have any other thoughts about the colored bar?

[Text box][applet]

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

The colored bar helped me create a stronger password.

[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.

Because of the colored bar, I created a different password than I would have otherwise.

[Strongly disagree to Strongly agree] (5-point scale)

Do you have any other thoughts about the colored bar?

[Text box][applet]
Which of the following statements best reflects how you entered your password on the previous screen?

- 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 these hints
- I stored my password for this study in a password manager or in my browser
- I wrote my password down on paper, and I type it in after looking it up
- I saved 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 password for this study, 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 took steps to memorize my password
- I wrote my passwords down on paper, and I type them in after looking them up
- I saved my passwords 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?

- I did not do any of the above
- I type my passwords in entirely from memory
- I write down or electronically store hints to help me remember my password for this study, and I type my passwords in after looking at these hints
- My passwords are automatically entered for me by a password manager or by my browser
- I saved 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 password for this study, and I type the password in after looking at the hint
- Other [text box input]

E.3 Part 2 survey

Which of the following statements best reflect 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 entirely from memory
- I had written my password down on paper, and I typed it in after looking at it
- I had saved my password electronically (e.g., in a file or on my phone), and I typed it in after looking at it
- 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 wrote my passwords down on paper, and I type it in after looking it 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 password for this study, but not my password itself
- I did not do any of the above
- Other [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.

**Table 6: Experiment 1 policy comparisons**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Peto-Peto</th>
<th>percent @ 10^6</th>
<th>percent @ 10^14</th>
<th>Noticed rejection</th>
<th>Creation time</th>
<th>Creation difficult</th>
<th>Creation annoying</th>
<th>Stored password</th>
<th>Part 1 dropout</th>
<th>Part 2 dropout</th>
<th>Recall time</th>
<th>Recall success</th>
<th>Remember difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocklists vs minimum-strength policies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c8 vs NN6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c8+Xato-strip-cifs</td>
<td>p = .005</td>
<td>p = .747</td>
<td>p = .001</td>
<td>N/A</td>
<td>p &lt; .001</td>
<td>p &lt; .001</td>
<td>p &lt; .001</td>
<td>p &lt; .001</td>
<td>p = .01</td>
<td>p &lt; .001</td>
<td>p = .678</td>
<td>p = .071</td>
<td>p = .163</td>
</tr>
</tbody>
</table>

**Table 7: Experiment 2 policy comparisons**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Peto-Peto</th>
<th>percent @ 10^6</th>
<th>percent @ 10^14</th>
<th>Noticed rejection</th>
<th>Creation time</th>
<th>Creation difficult</th>
<th>Creation annoying</th>
<th>Stored password</th>
<th>Part 1 dropout</th>
<th>Part 2 dropout</th>
<th>Recall time</th>
<th>Recall success</th>
<th>Remember difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocklists vs minimum-strength policies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c8 vs NN6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Minimum-strength and minimum-length requirement interactions |           |                |                  |                   |               |                    |                   |                 |                |               |             |             |                      |

| Blocklist and composition requirement interactions |           |                |                  |                   |               |                    |                   |                 |                |               |             |             |                      |
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 p-value parameter estimate and its 95% confidence interval are listed.

### Comparison Details

#### Table 8: Detailed statistical test results (1)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Xato-cifs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.84 (0.66, 1.08)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
</tbody>
</table>

#### Table 9: Detailed statistical test results (2)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Xato-cifs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.84 (0.66, 1.08)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
</tbody>
</table>

### Comparison Details

#### Table 8: Detailed statistical test results (1)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Xato-cifs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.84 (0.66, 1.08)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
</tbody>
</table>

#### Table 9: Detailed statistical test results (2)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Xato-cifs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.84 (0.66, 1.08)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
<tr>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs</td>
<td>Impact of bl. (1c8)</td>
<td>OR = 0.96 (0.79, 1.15)</td>
</tr>
</tbody>
</table>

### Summary

The tables present the results of statistical tests comparing different conditions or treatments. The odds ratio (OR) and its confidence intervals are provided for each comparison, along with the adjusted p-values.
### Table 1: Detailed statistical test results (I)

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests on password-creation annoyance (Chi-square, FET)</td>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.449</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.574</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>2.153</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.756</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>1.767</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.505</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.505</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.426</td>
</tr>
<tr>
<td>Tests on password storage after creation (Chi-square, FET)</td>
<td>Experiment 1</td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.699</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.937</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.724</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.037</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Xato-ciss Impact of h (1df)</td>
<td>0.184</td>
</tr>
<tr>
<td>Tests on password-dropping rate (Chi-square, FET)</td>
<td>Experiment 2</td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.699</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.937</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.724</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.037</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.184</td>
</tr>
</tbody>
</table>

### Table 2: Detailed statistical test results (II)

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Family</th>
<th>Details</th>
<th>Adj. p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests on recall success in Part 2 (Chi-square, FET)</td>
<td>Experiment 2</td>
<td>1c8+Xato-scifs v. 1c8+Pwned-fs Bl. reqs. for 1c8 Impact of bl. (1c8)</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8+Xato-scifs v. 1c8+Pwned-fs Bl. reqs. for 1c8 Impact of bl. (1c8)</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8+Xato-scifs v. 1c8+Pwned-fs Bl. reqs. for 1c8 Impact of bl. (1c8)</td>
<td>0.736</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3c8 v. 3c8+Xato-cifs Impact of bl. (3c8)</td>
<td>0.736</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3c8 v. 3c8+Xato-cifs Impact of bl. (3c8)</td>
<td>0.684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3c8 v. 3c8+Xato-cifs Impact of bl. (3c8)</td>
<td>0.044</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3c8 v. 3c8+Xato-cifs Impact of bl. (3c8)</td>
<td>0.044</td>
</tr>
<tr>
<td>Tests on password-reuse difficulty (Chi-square, FET)</td>
<td>Experiment 2</td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.176</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.179</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1c8 v. 1c8+Pwned-fs Impact of h (1df)</td>
<td>0.005</td>
</tr>
</tbody>
</table>

---

*Note: The tables and text are derived from the annotated pages, with statistical test results formatted according to the given guidelines.*
REFERENCES