Measuring Website Password Creation Policies At Scale

Suood Alroomi
Georgia Institute of Technology
Kuwait University
roomi@gatech.edu

Frank Li
Georgia Institute of Technology
frankli@gatech.edu

ABSTRACT

Researchers have extensively explored how password creation policies influence the security and usability of user-chosen passwords, producing evidence-based policy guidelines. However, for web authentication to improve in practice, websites must actually implement these recommendations. To date, there has been limited investigation into what password creation policies are actually deployed by sites. Existing works are mostly dated and all studies relied on manual evaluations, assessing a small set of sites (at most 150, skewed towards top sites). Thus, we lack a broad understanding of the password policies used today. In this paper, we develop an automated technique for inferring a website's password creation policy, and apply it at scale to measure the policies of over 20K sites, over two orders of magnitude (~135x) more sites than prior work. Our findings identify the common policies deployed, potential causes of weak policies, and directions for improving authentication in practice. Ultimately, our study provides the first large-scale understanding of password creation policies on the web.

CCS CONCEPTS

• **Security and privacy → Authentication**; *Web application security*; • **General and reference → Measurement**.

KEYWORDS

Online Authentication; Password Policies; Account Creation; Authentication Guidelines

ACM Reference Format:
Suood Alroomi and Frank Li. 2023. Measuring Website Password Creation Policies At Scale. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623156

1 INTRODUCTION

Passwords remain the de facto standard method for online authentication [6], and password-based web authentication will likely remain ubiquitous for the foreseeable future. As a consequence, the security of the web ecosystem is critically dependent on how both users and websites manage password authentication. Over the years, researchers have extensively explored how users behave with passwords, particularly when constrained by password creation policies (e.g., [21, 22, 24, 40–43, 46, 48, 50]). These efforts have produced insights into how authentication *should* be handled by websites to promote password security and usability. These user-centric efforts have helped drive significant updates to modern password guidelines, such as spurring the US National Institute of Standards and Technology (NIST) to release new online identity management guidelines [20] in 2017, its first since 2004 [9].

Ultimately though, websites are the entities that must implement recommended practices to improve authentication security and usability in reality. To date, there has been significantly less investigation into how website operators actually manage password authentication and what password creation policies they enforce. A handful of studies [7, 13, 15, 16, 23, 24, 27, 28, 35, 39, 49] have manually analyzed the password policies of top websites. However, due to the manual methods used in this prior work, the scale of investigation is heavily limited, with the largest entailing only 150 websites [7, 35]. The considered website populations also skew towards highly-ranked sites [7, 15, 16, 24, 28, 35, 39], across a few countries (i.e., US, Germany, and China) [28, 39, 49]. Furthermore, most studies were conducted over a decade ago [7, 13, 15, 16, 23, 27, 35], predating significant updates to password guidelines, including those by NIST [20] and Germany's Federal Office for Information Security (BSI) [18, 19]. Thus, we lack a large-scale modern understanding of the password creation policies deployed by sites today, the authentication security and usability implications of these policies, and the adoption rate of authentication recommendations.

In this work, we seek to close this gap. Given the incredible diversity of the web, doing so is challenging [24, 49], as websites and their password authentication are implemented in a myriad of ways, and password policy information is often not explicitly published. In this work, we develop a web measurement method that automatically infers password creation policies in a blackbox fashion. Our method entails testing specifically-chosen passwords in a carefully constructed order during a site's account signup, identifying which passwords are accepted or rejected to infer the site's password creation policy. We construct our inference method to reduce its footprint on evaluated sites. We apply our technique to successfully infer the password creation policies of over 20K websites across the Tranco top 1M, evaluating a diverse population over two orders of magnitude (~135x) larger than any prior study.

Our analysis reveals how often websites employ certain creation policy parameters, such as acceptable characters, character composition requirements, disallowed password structures, and breached password blocking. We find that the most common policies today enforce few requirements on passwords, aligning with recent policy recommendations (e.g., NIST's 2017 guidelines [20]). However, counter to modern standards, acceptance of short passwords is

† An extended version (EV) of this publication with additional appendices is available at https://arxiv.org/abs/2309.03384, omitted due to space constraints.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623156

widespread, with over half of sites allowing passwords of six characters or shorter, and an unexpected 12% lacking any minimum length requirements. Furthermore, 30% of sites do not support certain recommended characters in passwords, including spaces and special characters. We also observe only about 12% of sites using password blocklists, resulting in the majority of sites being vulnerable to password spraying attacks [29, 49]. Overall, only a minority of sites fully adhere to common guidelines, with most sites adhering to more dated guidelines. We also observe that top-ranked sites tend to support stronger policy parameters. Through case studies of weak policy parameters, we identify how web frameworks and default configurations may be driving factors.

Ultimately, our study illuminates the state of modern password creation policies at scale for the first time, while also highlighting authentication security and usability problems requiring attention and identifying directions for improving authentication in practice.

2 RELATED WORK

Here we summarize prior work measuring real-world password policies and studies that relied upon automated account creation.

2.1 Password Policy Measurements

Over the past 15 years, multiple studies have manually investigated the password policies used by real-world websites. Several initial studies [15, 16, 27] were very limited in scale (considering up to 10 sites). At a larger scale, Kuhn et al. manually surveyed the password policies of 69 domains in 2007 and then again in 2009 [23]. The authors noted that 45% of the websites changed their password policy in the two-year span. These changes included more widely imposing password complexity and length requirements, although policies on many sites remained weak. Similarly, in 2010, Florencio et al. explored the factors that influence the password policies employed by websites [13]. The authors manually characterized the password policies of 75 US websites, finding that factors related to monetization seemingly correlated inversely with policy strength. The study was replicated seven years later in 2017 by Mayer et al., using the same set of websites along with 67 additional German websites [28]. This work replicated the earlier observations, and observed that overall, password policies on US websites had increased in strength over time, and were stronger than those on German sites. In 2015, Wang et al. also compared the password policies between 30 Chinese websites and 20 English-language sites [49]. They observed several Chinese websites requiring digit-only passwords, and policies on English sites were overall more stringent.

At the largest scale, in 2010, Bonneau et al. conducted an extensive manual evaluation of the password policies on 150 domains chosen from the Alexa Top 500 sites [7]. They found that half of the websites enforced a minimum password length of 6, and 18% had no length restrictions. Furthermore, few sites disallowed common dictionary words for passwords. Due to password reuse by users across websites, the authors also highlighted the potential negative externalities caused by websites with weaker password policies, impacting the passwords chosen by users even on sites employing more secure policies. This concept was empirically explored further by Preibusch and Bonneau through a game-theoretic model using the same dataset [35]. In 2017, Seitz et al. also characterized the potential for password reuse across sites by contrasting the password policies across 100 German sites [39], finding that the policies were not diverse enough to mitigate the risk of password reuse. They were able to construct passwords that could be accepted across 99% of the sites. Most recently, Lee et al. [24] manually investigated 120 top English websites, finding that over half did not blocklist common passwords. Overall, less than a quarter of the sites followed security and usability password policy recommendations.

A primary limitation of these studies is that they manually analyzed website password policies. As a consequence, the studies were small-scale, with the largest involving only 150 sites, and the characterized sites heavily skewed towards top sites (summarized in Table 8 of Appendix A). Furthermore, most studies were over a decade ago, making their observations dated. The web has expanded significantly since then, and our understanding of secure password policies has also substantially evolved (including updates to modern authentication recommendations, such as NIST's latest password policy guidance released in 2017 [20]). Thus, a more modern view of website password policies is needed. Our study leverages automation to provide the largest-scale picture of web password creation policies today, encapsulating a diverse population of websites across different rankings.

2.2 Account Creation Studies

Several studies have used automated account creation for different measurements. DeBlasio et al. automatically created honey accounts on websites to detect potential credential theft [11]. They successfully created accounts across 2.3K websites, detecting 19 potential cases of website credential compromise. Recently, Drakonakis et al. investigated how websites handle cookies during authentication workflows [12]. They attempted automated account creation and login across 1.5M domains, successfully creating accounts on 25K domains in total. They found half of the domains vulnerable to cookie-hijacking attacks. While our automated account creation process shares similarities with the prior work, we designed our method from the ground up, as our end-to-end empirical method required overcoming distinct challenges, such as more extensive account creation activity and inferring password policies.

3 METHOD AND IMPLEMENTATION

Here, we describe our method for automatically inferring password policies. At a high level, we attempt multiple account signups on a website using different passwords, observing which accounts are successfully created to identify password policy parameters. As shown in Figure 1, we first discover a website's account signup workflow. To do so, we search for account signup forms (Section 3.2) across a website's pages to detect an account signup page (Section 3.3). Then, we execute our policy inference process, which attempts multiple account signups with different passwords (Section 3.4) while evaluating whether the signup is successful (Section 3.5). Based on which signup attempts (and the associated passwords) succeed, we infer the password policy parameters (Section 3.6). To conduct our measurements, we train two machine learning classifiers, one for signup form detection (Section 3.2) and another for classifying signup attempt success (Section 3.5). Other

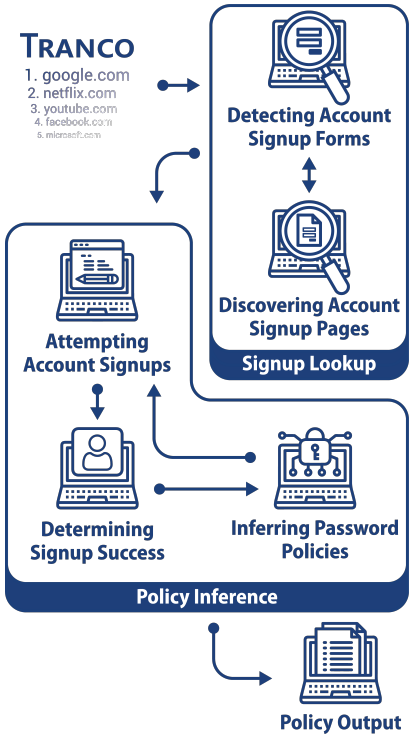


Figure 1: Illustration of the stages of our password policy measurement method.

components of our method rely on keyword-based heuristics (discussed in Section 3.1 and the Extended Version's (EV) Appendix A), particularly for identifying potential account signup URLs and form fields. We will share our measurement data and code to vetted researchers upon request, as otherwise these could potentially be used in online abuse.

3.1 Ground-Truth Analysis

Modern websites and their authentication workflows are diverse, in both design and implementation. As a consequence, we require heuristics throughout our method for discovering and analyzing website account creation (as have prior work conducting similar automated account creation [11, 12]). These heuristics include keywords for classifying webpages and HTML elements. We additionally train machine learning classifiers for complex labeling tasks.

To identify keywords for our specific method in a systematic, language-agnostic, and data-driven fashion, as well as to train our classifiers, we manually analyzed 2800 domains randomly sampled from the Tranco Top 1M [34] (from June 6, 2021). We identified whether each domain supports account creation (26% did), and if so, we analyzed the characteristics of its account signup workflow (including the location of its signup pages and forms). We refer to this dataset as our *ground-truth data*. For extracting relevant keywords, we applied keyword ranking algorithms to identify the top keywords prevalent in positive cases but uncommon in negative cases, agnostic to any specific language (details in EV Appendix A). We discuss training our classifiers in the following sections.

3.2 Detecting Account Signup Forms

To assess a site's password policies, we first identify its signup page and form. To distinguish account signup forms from others (e.g., login, newsletter), we use a binary SVM classifier. For its features, we use the presence of signup-related keywords (chosen from our ground-truth data, discussed in EV Appendix A) in the HTML form's title, ID, class, and action, as well as the numbers of form inputs in total and password-type inputs. For training data, we manually labeled the HTML forms in our ground-truth data. We trained our model using Python's sklearn [38], selecting hyperparameters using grid search. Evaluating our model with 10-fold cross-validation, we observe an average accuracy of 94.7% (errors discussed in EV Appendix C.1). Note that while false negatives cause us to skip evaluating sites, false positives result in unsuccessful attempts to evaluate them (which we detect and filter out).

3.3 Discovering Account Signup Pages

Given a domain, our method starts by searching for its signup page, identified by the presence of a signup form (from Section 3.2). This process proceeds as follows until a signup page is found.

(1) We search for a signup form on the domain's landing page.
(2) We next crawl URL links found on the landing page that contain common keywords for account signup or login URLs. We call these candidate URLs as they likely contain an account signup or login form. Keywords are selected using ground-truth data (see EV Appendix A), with separate keywords for signup and login URLs. We use login URLs as they often contain links to a signup page (for users without an account). On login URLs, we attempt to detect a signup form, otherwise we collect further candidate signup URLs (now ignoring candidate login URLs). For each page, we visit at most four candidate URLs to avoid excessively crawling a domain. (In our ground-truth data, we observed that this threshold was sufficient for discovering signup URLs, as most pages had few, if any, candidate signup or login URLs.)
(3) Finally, we query the Google search engine for the domain's account signup pages (using ScraperAPI [3]). Our search query includes the domain along with "account OR register OR sign+up OR create", constructed using the most frequent keywords in the HTML titles of real signup pages in our ground-truth data. Given the search results, we again consider candidate signup and login URLs, crawling up to 4 candidate URLs in search of a signup page/form (using the same method for identifying candidate URLs and processing them as done with URL links on the domain's landing page). (We observed that this crawling threshold was sufficient on our ground-truth dataset.)
(4) Here, we record the domain as lacking a signup page.

We note that our crawler is non-interactive and does not simulate user actions on a page. Some sites require an action for the signup form to fully appear (e.g., clicking a "signup" button, or clicking through multi-page forms). However, in our ground-truth data, this behavior is not widespread, and automating it would be challenging.

3.4 Attempting Account Signups

With a domain's signup page, we next fill out and submit the signup form. By testing different passwords across multiple signup attempts, we will infer the domain's password policy (discussed in

Section 3.6). Automatically filling and submitting a signup form encounters two key challenges.

First, we must identify signup form fields and provide acceptable values/actions. We classify them based on the HTML input element's name, class, and ID, using relevant keywords identified in our ground-truth data (see EV Appendix A). For common form fields (e.g., name, email), we use either pre-selected values (not real user data) or the Faker Python library [5] to generate synthetic data. We handle the password field specifically, as discussed in Section 3.6. For unrecognized fields, we generate a random string as a last resort. Some forms offer multiple button elements (e.g., signup and single sign-on buttons). We identify the account signup button using keywords derived from our ground-truth data (see EV Appendix A).

A second challenge is that many signup workflows require completing a CAPTCHA. In our ground-truth data, we identified CAPTCHAs on at least 49% of signup forms. We aimed to overcome CAPTCHAs to significantly increase our likelihood of successfully assessing sites. Given our measurement's scale and ethical concerns[1] with human-driven CAPTCHA solvers (discussed in Section 3.10), we opted to rely on an automated CAPTCHA solver, AZcaptcha[2] [2]. We identify CAPTCHAs during the signup process through fingerprinting the HTML/JavaScript code used by the CAPTCHA implementations supported by AZcaptcha, and pass the extracted CAPTCHAs to AZcaptcha to solve. (During our full measurement, AZcaptcha correctly solved 94% of all CAPTCHAs we encountered, with failure cases discussed in EV Appendix C.1)

3.5 Determining Signup Success

Websites vary widely in response to submitting an account signup form, and behavior differs depending on the signup success. For example, some sites redirect to another page, while others display a message. To determine if a signup attempt is successful, we develop an ensemble decision tree classifier that operates on features of the webpage returned upon form submission. We collected training data from signup attempts on 160 domains in our ground-truth data. Our features include the presence of a signup form (detected as in Section 3.2), keywords in the page and URL, and the similarity of the page and its URL with those before form submission. We then trained an XGBoost decision tree ensemble model with 100 trees, selecting hyperparameters using grid search. Evaluated using 4-fold cross-validation, we observe a 91.3% accuracy. Note that classification errors primarily result in consistent successes or failures across all attempts for a domain, which we detect and filter out.

3.6 Inferring Password Policies

The prior sections discussed our method for finding signup pages, as well as completing, submitting, and determining the submission outcome for the signup forms. To infer the password policy, we perform multiple signup attempts where we provide consistent signup information except we vary the passwords provided systematically, allowing us to determine the password policy parameters based on which passwords are accepted or rejected. We determine whether a password is accepted based on the form submission outcome. However, form submission may fail due to other information we provide, rather than just the passwords. In such cases, as we provide consistent signup information across signup attempts, we will observe consistent signup failures for a domain, independent of the passwords tested, and we can subsequently filter out such domains from our analysis. Also, a successful account signup results in a created account. To minimize the account-related resources we require of domains, we constructed our method to reduce the number of accounts created, as discussed further in Section 3.10.

3.6.1 Password Policy Parameters. We evaluate the following password creation policy parameters, which encapsulate all policy parameters investigated by prior work [7, 24, 39, 49], which fall into three classes. The first class involves password **lengths**:

- **Length** (L_{min}, L_{max}): The minimum and maximum password lengths allowed, respectively. We conservatively consider $L_{min} \in [0, 32]$ and $L_{max} \in [6, 128]$.

The second class of parameters is **restrictive**, as they require that all passwords exhibit certain character structure.

- **Digits** (DIG_{min}): The minimum number of digits required. We consider $DIG_{min} \in [0, 2]$.
- **Uppercase Letters** (UPP_{min}): The minimum number of uppercase letters required. We consider $UPP_{min} \in [0, 2]$.
- **Lowercase Letters** (LOW_{min}): The minimum number of lowercase letters required. We consider $LOW_{min} \in [0, 2]$.
- **Special Symbols** (SPS_{min}): The minimum number of special symbols required. We consider $SPS_{min} \in [0, 2]$.
- **Combination, 3 out of 4** (R_{cmb34}): Passwords must exhibit 3 out of 4 character classes (*Digits*, *Uppercase Letters*, *Lowercase Letters*, *Special Symbols*).
- **Combination, 2 out of 4** (R_{cmb24}): Passwords must exhibit 2 out of 4 classes (same classes as R_{cmb34}).
- **Combination, 2 out of 3** (R_{cmb23}): Passwords must exhibit 2 out of 3 classes (*Digits*, *Letters*, *Special Symbols*).
- **Combination of Words** (R_{2word}): Passwords must have multiple words, where a word is defined as a string of three or more letters (any case), delimited by digits or special symbols[3].
- **No Arbitrary Special Symbols** ($R_{no\_a\_sps}$): Passwords cannot have arbitrary special characters (considering less popular special characters not accounted for by parameters $P_{spn1} - P_{spn4}$).
- **Letter Start** (R_{lstart}): A password must start with a letter. (Prior work observed such positional restrictions [13].)

A final class of parameters is **permissive**, allowing certain password characteristics without requiring them.

- **Dictionary Words** (P_{dict}): Common dictionary words (e.g., `apple`) are permitted within the password, where a word is at least 3 letters.
- **Sequential Characters** (P_{seq}): Logical sequences of 3+ characters (e.g., `123`, `abc`) are permitted in the password.
- **Repeated Characters** (P_{rep}): 3+ consecutively repeated characters are permitted in the password.

[1] Prior automated account creation work skipped sites with CAPTCHAs [12] or used human CAPTCHA-solvers at a small scale [11].

[2] AZcaptcha [2] advertised an automated OCR-based method. We note that AZcaptcha's price point is also significantly lower than human-driven CAPTCHA solvers, reinforcing AZcaptcha's automation claims.

[3] We assume that if $R_{2word} = True$, then $L_{min} \geq 10$ (whereas in theory, L_{min} could be between 7 and 9). We argue that this is a reasonable assumption as requiring such word structure without allowing longer passwords would overly constrain user password selection, especially as the average word is 4.7 characters [32].

- **Long-Digit Passwords** (P_{longd}): All-digit max-length passwords are permitted (observed before on Chinese websites [49].)
- **Short-Digit Passwords** (P_{shortd}): All-digit min-length passwords are permitted (used along with P_{longd} to determine length's role in accepting digit-only passwords)
- **Personal Information/Identifiers** (P_{id}): Personal information (e.g., username) is permitted in the password.
- **Space** (P_{space}): Whitespaces are permitted in the password.
- **Emojis** (P_{emoji}): Emojis are permitted in the password.
- **Unicode Letters** (P_{unicd}): Unicode characters (e.g., accented characters) are permitted in the password.
- **Popular Special Symbols** (P_{spn1}–P_{spn4}): The four most popular special symbols (".", "!", "_", and "#", respectively) are permitted in the password. We derive this list of top special symbols by analyzing 10M passwords in a popular password dataset [1].
- **Breached Passwords** (P_{br}): A common password from a known password leak is permitted.

3.6.2 Inference Algorithm. With many parameters to infer, we require an efficient algorithm that evaluates a limited number of test passwords. We describe our algorithm here, with further details (including correctness and efficiency) in EV Appendix B.

Algorithm Steps. At a high level, our inference algorithm operates by first finding one acceptable password (chosen in a specific fashion). Then, we evaluate each policy parameter one by one, testing passwords that are modifications of the original admissible password where only the specific parameter's dimension is changed, to determine that parameter's value. The order of parameter evaluation is specifically chosen to isolate the impact of just that parameter and minimize the number of successful account signups. Concretely, our algorithm operates in five steps.

Step 1. Admissible Password: First we must find an admissible password to seed our exploration, which satisfies the restrictive parameters (e.g., minimum class requirements) and all permissive parameters (e.g., avoiding the relevant password characteristic such as repeated letter and number sequences.)

For a given length l, we identify that there exists only a small set of passwords (which we call the *safe set*) for which one password will satisfy any possible parameter combination. If a website accepts passwords of length l, then the safe set must contain at least one acceptable password.

While we consider a variety of parameters, the safe set is small because a password can satisfy multiple restrictive parameters simultaneously (e.g., contain multiple characters of all classes, satisfying all minimum class and class combination parameters), and also satisfy all permissive parameters by avoiding the relevant password characteristic (i.e., avoiding certain characters and sequences).

We manually construct the safe sets for lengths $l \in [6, 32]$, shown in Table 1, covering the range of lengths that we conservatively assume a site must accept (based on our L_{min} and L_{max} assumptions). As seen in Table 1, the safe set for a given length contains passwords covering all restrictive parameter combinations, while also satisfying all permissive parameters. Note that for short lengths, fewer restrictive parameters can be concurrently satisfied, so the safe set is larger. The largest safe set contains 10 passwords (for $l = 6$), while for lengths 8 or larger, the safe set consists of only two passwords (with and without special characters).

Password	L	$R_{no\_a\_sps}$	LOW	UPP	DIG	SPS
M-7c4@	6		1	1	2	2
M-7cS@			1	2	1	2
Mx-7c@			2	1	1	2
Mx7c4@			2	1	2	1
Mx7cS@			2	2	1	1
M-7cS4			1	2	2	1
M-7S4@			0	2	2	2
x-7c4@			2	0	2	2
Mx-cS@			2	2	0	2
Mx7cS4		T	2	2	2	0
M7-cS4@	7		1	2	2	2
Mx7-c4@			2	1	2	2
Mx-cS4@			2	2	1	2
Mx7-cS4			2	2	2	1
Mx7zcS4		T	2	2	2	0
Mx7-cS4@	8		2	2	2	2
MxT7zcS4		T	2	2	2	0
Mx7-cS4@y	9		2	2	2	2
MxT7zcS4t		T	2	2	2	0
MxT7zcS4-@	10		2	2	2	2
MxT7zcS4t1		T	2	2	2	0

Table 1: The safe set of passwords for different lengths L. For each password, we indicate which restrictive parameter configurations are satisfied. Note that all passwords satisfy the class combination parameters, R_{lstart}, and R_{2word} (if $L \geq 10$). Permissive parameters are also all inherently satisfied. For $L > 10$, the safe set is identical as with $L = 10$, except with passwords padded with arbitrary letters and digits to length.

We search for an admissible password through the safe sets in increasing length order, first testing passwords with special characters within each safe set. Whether the admissible password found contains a special character already determines our first restrictive parameter $R_{no\_a\_sps}$ (if arbitrary special characters are disallowed). In subsequent steps, we modify this admissible password along a single parameter's dimension and identify whether the modified password remains accepted, revealing the parameter's value.

Step 2. Restrictive Parameters: With an admissible password of length l (and $R_{no\_a\_sps}$ determined, which indicates whether arbitrary special characters are allowed), we then evaluate the restrictive parameters first, as determining these reveal the constraints enforced on any further tests. To determine the value of a restrictive parameter, we modify the admissible password to only violate that parameter, observing whether the modified password is accepted. If so, then the restrictive parameter is in effect.

1) Combination of Words (R_{2word}): If $R_{2word} = True$, the admissible password must contain a two-word structure, delimited by a non-letter character (if not, then we already know $R_{2word} = False$). To test R_{2word}, we modify the admissible password by moving the non-letter delimiter to the password end, eliminating the two-word structure (e.g., Admissible Password: MxT7zcS4-@, Modified Password: MxTzcS4-@7). If this modified password is no longer accepted, $R_{2word} = True$, otherwise $False$. This modification does not affect other parameters as the length and character composition remain identical, and there are no other positional restrictions on middle-of-password characters. Permissive parameters are also not affected as the modification does not introduce a character sequence related to a permissive parameter (e.g., sequential/repeated characters, dictionary word).

2) Letter Start (R_{lstart}): All our admissible passwords begin with a letter. To assess R_{lstart}, we move the first non-letter character in the admissible password to the start (e.g., Admissible Password: Mx7-cS4@, Modified Password: 7Mx-cS4@). If accepted, $R_{lstart} = False$, otherwise $True$. If $R_{2word} = True$, we take care to avoid moving the two-word delimiter (e.g., Admissible Password: MxT7zcS4t1, Modified Password: 4MxT7zcSt1), as all admissible passwords have multiple non-letter characters (see Table 1). This modification does not affect other parameters as the length and character composition remain identical, and the only other positional restriction remains satisfied. Also, moving the non-letter characters does not introduce a character sequence affecting a permissive parameter.

3) Character Class Minimums (DIG_{min}, UPP_{min}, LOW_{min}, SPS_{min}): To find the character class minimum for class C (where C is either digits, uppercase letters, lowercase letters, or special symbols), we modify the admissible password to contain no C characters, by replacing C characters with characters of other classes (e.g., if $C = LOW$, Admissible Password: Mx7-cS4@, Modified Passwords: MX7-CS4@). If accepted, $C_{min} = 0$. Otherwise, we modify the admissible password to contain only one C character (e.g., if $C = LOW$, Admissible Password: Mx7-cS4@, Modified Passwords: MX7-cS4@). If accepted, $C_{min} = 1$, otherwise $C_{min} = 2$.

To avoid conflicting with other restrictive parameters, our default replacement policy is to swap between lowercase and uppercase characters (to not impact R_{2word} and R_{lstart}), and between digits and special symbols (to not affect R_{2word}). If $R_{no\_a\_sps} = True$ (no special characters allowed), digits are instead replaced with any letters (note here that if $R_{2word} = True$, then $DIG_{min} \geq 1$).

In most cases, all class combination parameters (R_{cmb23}, R_{cmb24}, R_{cmb34}) remain satisfied without further consideration. As seen in Table 1, most admissible passwords already have four character classes, so three classes remain after eliminating one class in the admissible password. A few admissible passwords have only three character classes (none have fewer classes), either because they are short (specifically, $l = 6$) or because $R_{no\_a\_sps} = True$ (so only three classes are allowed). For $l = 6$ admissible passwords, there are two characters of each class, and we can replace the second C character with one from the missing class, following the default replacement policy for the first character (e.g., if $C = UPP$, Admissible Password: Mx-cS@, Modified Password: mx-c1@). This preserves R_{lstart} while maintaining 3 distinct classes. When $R_{no\_a\_sps} = True$, the class combination parameters either implicitly imply class minimums which we will correctly infer (e.g., $R_{cmb34} = True$ means there needs to be one character of each class), or will remain satisfied (the modified password still has two classes).

4) Combinations Requirements (R_{cmb23}, R_{cmb24}, R_{cmb34}): To evaluate the final set of restrictive parameters, the class combination requirements, we modify the admissible password to have fewer classes and test for acceptance.

We start by identifying required character classes based on the other restrictive parameters. R_{2word} and R_{lstart} both require letters; we select the required case based on class minimums, selecting lowercase letters by default. Similarly, R_{2word} requires either digits or special characters; we select which based on class minimums and $R_{no\_a\_sps}$, selecting digits by default.

For modifying our admissible password, we replace all characters of non-required classes with those of a required class (replacing with lowercase letters if no class is required). If letters are required at certain positions, we replace any letters of a non-required class with letters of the required class (likewise between digits and special characters). This modified password has the minimum number of classes while adhering to other restrictive parameters, without impacting length or permissive parameters (e.g., if $UPP_{min} \geq 1$, Admissible Password: Mx7-cS4@, Modified Password: MXZNCSZA). If the modified password is accepted, we can determine the class combination parameters given the required classes in the password (in the prior example, there are no class combination requirements).

However, if not accepted, then an explicit class combination requirement is in effect. We determine its configurations based on the properties of the rejected modified password, as follows:

- *All non-letters of one class (e.g., if $DIG_{min} \geq 1$, $R_{lstart} = False$, Admissible Password: Mx7-cS4@, Rejected Modified Password: 32729041).* Here, the other restrictive parameters require a single non-letter class. We test a new modification of the admissible password with only that non-letter class and letters of one case, using lowercase by default (e.g., New Modified Password: a2729041). If this new password is accepted, $R_{cmb23} = R_{cmb24} = True$ (and $R_{cmb34} = False$), otherwise only $R_{cmb34} = True$.
- *All non-letters of both classes (e.g., if $DIG_{min} \geq 1$, $SPSmin \geq 1$, $R_{lstart} = False$, Admissible Password: Mx7-cS4@, Rejected Modified Password: 157-824@).* Here, we can immediately infer that only $R_{cmb34} = True$ as a two-class password was rejected.
- *All letters of one class/case (e.g., if $UPP_{min} \geq 1$, Admissible Password: Mx7-cS4@, Rejected Modified Password: MXZNCSZA).* We test a new modified password with letters of both cases (e.g., New Modified Password: MxZNCSZA). If accepted, only $R_{cmb24} = True$. If not, move to the following case.
- *All letters of both classes/cases (e.g., if $UPP_{min} \geq 1$, Admissible Password: Mx7-cS4@, Rejected Modified Password: MxZNCSZA).* If both letter cases are required, we know $R_{cmb23} = R_{cmb34} = True$. Otherwise, we test a new modified password with letters of only one case (whichever is required, defaulting to lowercase letters) and digits (e.g., New Modified Password: M3ZNCSZA). If accepted, only $R_{cmb23} = True$, otherwise only $R_{cmb34} = True$.
- *Contains one non-letter class and one letter-class (e.g., if $UPP_{min} \geq 1$, $DIG_{min} \geq 1$, Admissible Password: Mx7-cS4@, Rejected Modified Password: MX71CS41).* Here, we can immediately infer that only $R_{cmb34} = True$ as the two-class password was rejected.

Step 3. Length Parameters: Having now determined the restrictive parameter values that constrain password structure, we can construct passwords of different lengths that satisfy the restrictive parameters (while implicitly satisfying all permissive parameters by avoiding associated characters and sequences). We can then determine the password length minimum and maximum through using binary search to test the acceptance of passwords of varying length (within the ranges $L_{min} \in [0, 32]$ and $L_{max} \in [6, 128]$). For example, to evaluate L_{max}, we first construct and test a password of length 67 (halfway point of our range). If accepted, we recursively explore L_{max} within the upper half [68, 128], otherwise we explore the lower half [6, 66]), following the logic of binary search.

We detail our password construction algorithm in EV Appendix B. At a high-level, the restrictive parameters provide a set of required characters and positional constraints, and we satisfy these constraints first before adding additional characters to construct

a password of an evaluated length l. We start constructing a password using characters required by the class minimums, then using characters of other not-yet-used classes to satisfy class combination requirements (adhering to $R_{no\_a\_sps}$). If R_{lstart} and/or R_{2word} are true, we satisfy these positional constraints at the start of the password, again first using allowed characters of classes required by the class minimums and combination requirements (and any remaining required characters are added after the positional constraints). At this point, our partially-constructed password is the shortest that satisfies all restrictive parameters. If its length already exceeds the evaluated length l, we consider l an unacceptable length. Otherwise, we pad the password with arbitrary letters and digits to length l (e.g., if $UPP_{min} = DIG_{min} = 1$ and other restrictive parameters are false, Constructed Length-11 Password: M7ak3jCbE43).

Step 4. Permissive Parameters: Next, we determine the permissive parameters (i.e., what is allowed in passwords). To do so, we inject the character(s) associated with a permissive parameter (e.g., emoji, dictionary word) into an admissible password, while still satisfying restrictive, length, and other permissive parameters, and test if the modified password is accepted. If so, then the permissive parameter is true, and the associated characters are permitted.

1) Permitted Characters (P_{space}, P_{unicd}, P_{emoji}, P_{spn1} - P_{spn4}): We first generate an admissible password of maximum length (described in Step 3). We then test a modified password where a non-essential character (i.e., one not used to satisfy a restrictive parameter) is replaced with the evaluated character (if not possible, then the parameter value is inherently false) (e.g., for P_{spn4}, Generated Password: Mx7-a1p5b2, Modified Password: Mx7-a1p5b#). For P_{space}, we require that the whitespace character is not at the start or end of the password. This modified password remains adherent to restrictive, length, and other permissive parameters. If accepted, the permissive parameter value is true.

2) Permitted Sequences (P_{rep}, P_{seq} , P_{dict}, P_{id}): Here, we construct a password with the evaluated sequence and test for acceptance. For repeated characters (P_{rep}) the sequence is three repeating consecutive characters (e.g., 111, aaa, or AAA), and for sequential characters (P_{seq}) it is abc, 123, or ABC. For both parameters, we select one as permitted by other policy parameters.

For dictionary words (P_{dict}), we identify the longest word (up to 8 characters) permitted in a password as constrained by other policy parameters, and test the inclusion of the most common English word [36]. For personal identifiers (P_{id}), the evaluated sequence is a subset of the username used during account creation. We choose our usernames to be a 3-letter names followed by 5 random digits, and the sequence is the 3-letter portion of the username (e.g. if the registered username is joe31426, we evaluate the acceptance of the sequence "joe" in the password)

We first construct the shortest password P that satisfies the restrictive requirements (as done in Step 3). If the evaluated sequence can be added to the end of P while remaining within L_{max}, we simply test this augmented password, padding if necessary to reach L_{min} (e.g., to test for P_{seq} if $L_{min} = 6$, $L_{max} = 64$ and the shortest password satisfying restrictive parameters is: AQ16-@, Modified Password: AQ16-@abc). This augmentation does not affect restrictive parameters (nor length and other permissive constraints).

However, it is possible that appending the sequence to P does not fit within L_{max}. In such cases, P must already be near L_{max}-length (as we only require appending 3 characters). Instead, we must construct the evaluated sequence using characters already existing in P. We find the most common class C in P amongst lowercase letters, uppercase letters, and digits (for P_{dict} and P_{id}, we only consider the two letter classes). We then rearrange the characters in P to cluster C characters together. If three (or more) C characters are consecutive, we replace them with the evaluated sequence. Otherwise, we add the C characters necessary to form a 3-C-character substring, again replacing this with the evaluated sequence. By using the most common class, we minimize the additional characters that may need to be added (e.g., to test for P_{seq} if $L_{max} = 8$ and the shortest password satisfying restrictive parameters is: AQ16-@, Modified Password: ABC16-@). If the password cannot be constructed within length L_{max}, it is inherently false.

If restrictive parameters do not specify positional constraints, the rearrangement of P's characters does not violate any restrictive parameters (nor length or other permissive parameters). If R_{lstart} or R_{2word} specify positional constraints, we handle each specifically. We ensure that the rearranged password starts with a letter if $R_{lstart} = True$. If $R_{2word} = True$, then P contains a two-word structure, which must have at least 3 characters of one letter class. We cluster three letters of this class as one of the 3-letter words, and replace it with the evaluated sequence.

3) Long and Short Digit Passwords (P_{longd}, P_{shortd}:) We generate digit-only passwords of lengths L_{max} and L_{min}, respectively (without sequences/repetition). Here, restrictive parameters are ignored to explore exceptions for all-digit passwords (e.g., if $L_{min} = 6$, Attempted Password: 147036).

4) Breached Passwords (P_{br}): With other parameters determined, we test the highest-ranked breached password [8] satisfying them (e.g. if $DIG_{min} \geq 1$, $R_{lstart} = False$, $L_{min} \geq 6$, all other restrictive parameters are false and permissive parameters are true, Attempted Password: 123456, the most popular password in [8] satisfying policy parameters). If accepted, $P_{br} = False$, otherwise true.

Step 5. Sanity Check: Given an inferred policy, we test one final password that should not succeed (e.g., too short, violates restrictive parameters), as a sanity check. A detected success indicates a policy inference error, which we can filter out. (We also filter out other errors, where all attempts are successes or failures, and those where trailing attempts all fail, as discussed later.)

Algorithm Efficiency. Our algorithm systematically evaluates a website's password policy in an efficient fashion that avoids brute-force guessing passwords. As we can pre-compute the safe sets for our full range of explored lengths, and all policy parameters have a limited range of values (including length, which is efficiently investigated through binary search), we can determine the bounds on the number of passwords tested, as well as the bounds on the number of successful passwords accepted by a website. Table 2 depicts these bounds for each step of our inference algorithm, as well as for the entire algorithm. In the worst-case, our method will create up to 37 accounts on a website, with at most 105 account signup attempts (in most cases, the number of attempts and accounts created is significantly lower). We note that we prioritized fewer accounts created, as the impact of a failed account signup attempt on a website is much lower. Also, there is precedence in the research community

Algorithm Step	# Attempts	# Successes
Step 1: Admissible password	[1, 65]	[1, 1]
Step 2: Restrictive parameters	[4, 13]	[0, 9]
Step 3: Length parameters	[11, 12]	[0, 12]
Step 4: Permissive parameters	[2, 14]	[0, 14]
Step 5: Sanity check	[1, 1]	[0, 1]
Total: Whole algorithm	[19, 105]	[1, 37]

Table 2: Bounds on the number of account signup attempts and successes required by our method, per domain.

for creating test accounts for measurement purposes; existing studies on password policies also created multiple accounts to evaluate policy parameters, but did so manually [7, 13, 24, 28, 35, 39, 49].

Algorithm Correctness. EV Appendix B describes how each parameter is correctly evaluated in isolation. To further ensure correctness, we tested our inference algorithm on a thousand randomly-generated valid policies, observing only correct inferences.

3.7 Measurement Implementation

We implement our measurement method using Selenium browser automation [4] with headless Chrome instances[4] To minimize the computational load we induce on websites, as well as avoid triggering anti-bot detection, we rate limit our crawling of a domain to at most one page load every 30 seconds, and at most one account signup attempt every 30 minutes. We also use a pool of 14 proxies, switching to a new proxy for each signup attempt to provide IP diversity. Given the rate limiting, we highly parallelize our analysis across sites, such that sites are assessed in a round-robin fashion.

3.8 Limitations

Our measurement method is best-effort, relying on multiple heuristics. It can exhibit false negatives, missing some sites with account signups, such as those with complex workflows (e.g., multi-page forms), user verification (email or phone) prior to signup form submission, registration fees, or offline membership (details in EV Appendix C.1). Furthermore, our evaluation may fail on sites that can detect our measurements (e.g., sites deploying anti-bot defenses) or where our machine learning models misclassify. However, as our method follows a consistent workflow for account signup attempts, we can filter out errors where all attempts are detected as successful or failures, which is infeasible, as well as those where trailing attempts are all failures (as this is highly unlikely, as discussed in Section 4). Also, our final method step involves testing the inferred policy, further reducing the likelihood of false positives.

Our measurements also assume static policy parameters, rather than dynamic rules, such as if a site were to enforce password strength requirements. To evaluate whether password strength enforcement occurs at scale, we calculated the strength of all accepted passwords on successfully evaluated sites using password strength estimator `zxcvbn` [52]. We observe that for 94% of sites, the weakest accepted password was rated 2 or lower (out of 4), which is considered a relatively weak password (ranging from "too guessable" to "somewhat guessable"). Thus, it is unlikely that most sites are enforcing high password strength requirements.

Due to our method limitations, our evaluated sites may skew from domains with complex or unique workflows, as our analyzed domains use single-step account creation workflows, specific common keywords, and do not require verification or payment for signups. While our work does not comprehensively evaluate all sites (similar to all prior automated account creation works, including those investigating authentication [11, 12]), our dataset (discussed in Section 4) is orders of magnitude larger and more diverse (including across rankings) than prior studies, serving as more generalizable empirical grounding. Furthermore, as detailed in EV Appendix C.3, we manually investigated the password policies of a random sample of domains that our method does not handle, and found that our study's core findings generalize to these domains.

3.9 Alternative Measurement Approaches

While our automated account creation process is similar to prior work [12], our task involves distinct challenges (e.g., password policy inference), so we designed our method in a data-driven fashion from scratch. In comparison, while prior work applied rule-based heuristics for keyword selection, form detection, and verification, we applied machine learning techniques for such tasks. Our signup discovery process also uses search engine results to improve discovery. Our efforts resulted in effective account creation automation, even compared to prior work (see EV Appendix C.2).

We initially explored non-blackbox methods for assessing password policies, which could reduce website interactions. However, we manually evaluated a random sample of 200 signup websites and identified significant limitations.

Mining Textual Policy Descriptions: Only 25% of sampled sites provided policy descriptions (prior work observed 22% [7], as well as inconsistencies between policies and their descriptions [24]). Such descriptions are also diverse, often displayed only upon user action, and require natural language processing, yet often still do not describe all policy parameters (e.g., password blocklisting).

Inspecting Client-Side Policy Checks: Only 10% of sampled sites had client-side JavaScript password policy checks, which were custom implemented per site, inhibiting automated analysis.

Analyzing Strength Meters: Only 11% of sampled sites displayed password meters (recent work found only 19% on top English sites [24]). Prior work has also observed widespread custom meter designs [47], inhibiting automated analysis. Furthermore, sites typically use password meters as nudges instead of enforcing strength requirements [16, 49], and various policy facets (e.g., blacklisting, allowed characters) may not be factored into strength meters.

Using Password Resets: One might assess password policies through password reset workflows. However, we did not log into accounts to avoid account activity (as discussed in Section 3.10). Furthermore, many sites prevent choosing a new password similar to previous ones, which would interfere with policy inference. Finally, sites exhibit diverse password recovery workflows, often requiring user verification, complicating automated analysis.

[4]When crawling with a headless browser, websites may detect and block such a crawler. However, when debugging our method, we tested full browser instances and did not observe higher crawling success, likely because many sites either do not block crawlers or apply anti-bot techniques that are similarly effective on full browsers.

3.10 Ethics

As our study involves evaluating a large number of websites, there are several important ethical considerations. It is impractical to obtain consent from all sites. Furthermore, obtaining consent could negatively impact the scientific validity of our study, as websites may opt-out in a biased manner, may change their policies in light of our investigation, or may specifically block our measurements. Thus, we do not seek consent from the studied sites, and must carefully design our measurement methods. We extensively explored various measurement methods (as detailed in Section 3.9). Here, we discuss the concerns with our resulting approach, the potential harm associated with our study, and our mitigations.

To assess the password policies on websites, we attempt multiple account signups in an automated fashion, succeeding for some attempts. Prior studies have performed similar automated account creation [11, 12], and we draw inspiration from their ethical considerations in designing our method. The potential harm that this activity causes for websites includes the computational resources incurred by the website in processing our signup attempts and created accounts. To limit the resources that websites must expend due to our study, we constructed our password inference algorithm to reduce the number of attempts and successful accounts registered. For successfully created accounts, we never access, verify, or use those accounts. We also crawl websites and attempt account signups in a heavily rate-limited fashion, ensuring that a website receives at most one attempt every half hour (and in most cases, attempts occur even less frequently). We believe that for websites supporting account registrations, this rate of signup attempts and the number of accounts created requires a limited amount of storage and load on websites, and should not tax even small websites. Furthermore, there is precedence in the research community for creating small numbers of test accounts for measurement purposes; existing studies on password policies also created test accounts to evaluate policy parameters, but did so manually [7, 13, 24, 28, 35, 39, 49] (e.g., Seitz et al. [39] created up to 15 accounts per site). As part of our account creation method, we solve CAPTCHAs using an automated CAPTCHA solver. We avoid human-driven CAPTCHA solvers due to ethical issues identified with such services [30].

From the legal perspective, we consulted our organization's general counsel, as our methods may be contrary to some websites' policies and terms of services, which we are unable to explicitly check for all sites in our study. General counsel reviewed this study and determined that the legal risk is minimal, with support from judicial precedence, and that there lacked damages incurred by websites. Our organization's administration also reviewed and approved this study. Finally, there are no human subjects concerns with this study (as such, we were not reviewed by our organization's Institutional Review Board). No real user data was used for this study, and our study did not interact with any individuals.

4 RESULTS

Here, we apply our measurement method to evaluate the password policies of websites in the Tranco Top 1M. We analyze the top password policies, the values of the various policy parameters, adherence to modern guidelines, and differences across rankings.

Rank	Policy	%
1	$L_{min} = 1$	8.3
2	$L_{min} = 6$	7.1
3	$L_{min} = 5, L_{max} = 40$	4.1
4	$L_{min} = 8$	3.4
5	$L_{min} = 5$	2.9
6	$L_{min} = 12$	2.8
7	$L_{min} = 4$	1.2
8	$L_{min} = 8, R_{cmb34} = T$	0.8
9	$L_{min} = 8, L_{max} = 72$	0.8
10	$L_{min} = 7$	0.7
11	$L_{min} = 4, L_{max} = 40$	0.5
12	$L_{min} = 8, P_{longd} = F, P_{shortd} = F$	0.5
13	$L_{min} = 8, LOW_{min} = UPP_{min} = DIG_{min} = 1$	0.4
14	$L_{min} = 4, L_{max} = 20$	0.3
15	$L_{min} = 6, L_{max} = 100, P_{emoji} = F$	0.3

Table 3: Top 15 password policies for all evaluated sites. For each policy, unless specified otherwise, $L_{max} = 128$, minimum required characters of a class is 0, restrictive parameters are false, and permissive parameters are true.

4.1 Aggregate Measurement Results

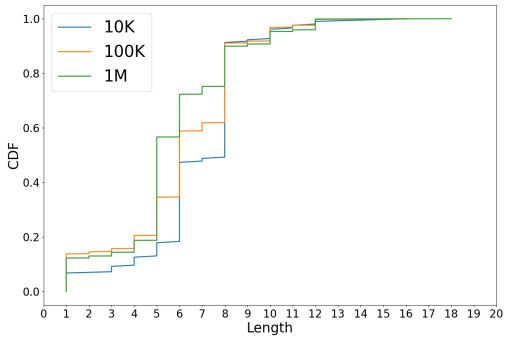
We conducted our large-scale measurement in Dec. 2021, evaluating password policies across Tranco Top 1M (Dec. 13). Appendix Figure 3 visualizes the site population at each method stage.

Out of the 1M domains, we find signup pages on 141K domains (14.12%). While we could successfully submit one signup attempt (including CAPTCHA solving) on 59K domains, we were able to fully evaluate (across multiple attempts) 26K domains. Finally, we filter out domains where all signup attempts are reported as successes or failures (as this is not feasible, especially with our sanity check signup attempt), or where all trailing attempts are failures (we test permissive parameters last, and as discussed shortly, it is highly unlikely that any site truly does not permit all tested characters/structures). This filtering leaves us with 20,119 domains for which we successfully analyze password policies. We manually validated our results are accurate on a random sample of 100 evaluated sites. We note that this population is two orders of magnitude larger than prior work (as discussed in Section 2), providing large-scale data on password policies for the first time.

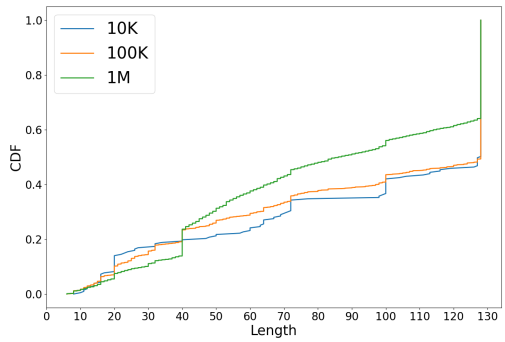
Our analyzed sites are also broadly distributed across rankings (unlike prior work's focus on top sites), with a slight skew towards lower-ranked sites, as shown in EV Appendix Figure 3. Across each 100K ranking interval, our final dataset contains between 1.4K–3.7K sites (and between 12.1K-19.2K signup sites found). In the subsequent discussion of our results, we separately consider our evaluated sites that are within the top 10K, 100K, and 1M (full dataset). Here, our results for Top X sites represent only the domains that we evaluated within the Top X ranking, rather than all Top X sites (as we did not evaluate all sites).

4.2 Top Policies

To start, we group websites with identical password policy configurations (across all policy parameters), and consider the top password policies observed among our websites. Table 3 lists the top 15 policies observed across our 20K websites (spanning the



(a) Minimum password lengths



(b) Maximum password lengths

Figure 2: CDFs of password minimum and maximum length requirements, for all sites in our dataset (Top 1M) as well as those ranked in the top 10K and 100K.

Tranco top 1M sites), and the percent of sites using those policies. Among the top policies, the majority (11 of 15) are simple policies, only constraining the password length without further restrictions. Surprisingly, the most popular policy (8.3% of sites) allowed passwords of any length without any constraints. Such a policy allows even single character passwords (we manually verify this behavior on a sample of sites), which are extremely weak passwords. Other top policies allow short passwords (e.g., 4, 5, and 6 characters). In addition, 5 top policies also cap the password's length (including one that limits passwords to only 20 characters). Other password constraints are less prominent in top policies, with only 4 of the top 15 policies applying any non-length constraints.

We find that policy popularity among sites exhibits a long-tail distribution. While the most popular policy was seen on 8.3% of sites, the top 10 policies cover only 32.1% of sites, with a total of 11,184 distinct policy configurations. Most policies appear on only one site, which highlights enormous diversity in the policies deployed (with implications for guidelines, password usability, and password managers, as will be discussed in Section 6).

4.3 Policy Parameters Values

Here, we evaluate individual password policy parameters. As the top 15 policies (Section 4.2) capture only a third of our sites, their parameters do not necessarily reflect an aggregate perspective.

		Lower	Upper	Digit	Special
0	10K	78.4	81.7	71.2	80.3
	100K	79.2	79.4	76.3	82.5
	1M	84.1	83.7	82.0	86.3
1	10K	10.6	10.1	20.7	14.9
	100K	11.6	10.9	14.5	9.8
	1M	8.3	8.7	10.0	7.0
2	10K	11.1	8.2	9.7	9.2
	100K	9.2	9.7	9.2	7.7
	1M	7.5	7.6	8.0	6.8

Table 4: For different character classes, we list the percent of sites in the Tranco Top 10K, 100K, and 1M (full dataset) that require a certain number of characters of that class.

4.3.1 Length. Figure 2a plots the CDF of the minimum password lengths enforced by password policies across our websites (Top 1M). As also seen with top policies, we find that a non-trivial fraction of sites (~12%) allow single-character passwords. The most prevalent minimum length is 5, seen at nearly 40% of sites. Only 25% of sites require passwords of length 8 or longer, as recommended by most modern guidelines [10, 19, 20, 31, 37], and ~10% require 10+ lengths.

Figure 2b similarly depicts the CDF of the maximum password lengths allowed by our websites. We observe that 36% of sites do not cap the password length (or allow at least 128 characters). The most common cap was 40 characters, observed at about 10% of sites. For other sites, the maximum length widely varied, although we notice prevalent use of lengths 20, 72, and 100. Overall, nearly 60% of sites allowed passwords of at least 64 characters, as recommended by many current guidelines [10, 20, 37]. We also find that a small portion of sites (1.7%) do not allow passwords longer than 10 characters, which is shorter than some sites' *minimum* lengths.

Case Study: $L_{min} = 1$**.** We manually investigated 475 detected sites and verified the correctness of our measurements. Through analyzing the JavaScript libraries and links embedded on these sites, we identified that the common pattern exhibited was simply accepting any non-empty password field, without applying password length logic. Interestingly, while this logic was customized for the majority of sites, we observed the prevalence of several web frameworks across these sites that we manually confirmed do not support password length constraints by default, such as WooCommerce (19% of such sites) and XenForo (1%).

Case Study: $L_{min} = 5$**.** We investigated the most common minimum length of 5 (38% of sites). Manually investigating a sample of 500 domains, we found 85% using the Shopify platform. We confirmed with Shopify customer support that their default length minimum was 5, indicating the influence a platform can have.

4.3.2 Restrictive Parameters. In Table 4, we display the percent of sites requiring a minimum number of class characters, for each character class. We see that the vast majority of sites (82-86%) do not enforce such requirements, with special characters being least likely to be required and digits being most likely. Of the remaining sites that do, approximately half require one character of a class, while another half require two (or more). We note that higher numbers of required characters of a class increase the complexity in creating passwords, which prior research has demonstrated can ultimately

diminish the security and usability of passwords [41], and is no longer recommended by many guidelines [20, 31, 37].

Similarly, Table 5 lists the prevalence of the remaining restrictive requirements. Derived from these results, we observed a similar prevalence of character class combinations (15% of distinct sites have at least one required combination, considering all combination possibilities) as with character class minimums (with 11% of sites using both character class minimums and class combination requirements). Furthermore, as seen in Table 5, we note that a non-trivial portion of sites (2.4%) require word structure in passwords, while 2.9% of sites require passwords to begin with a letter. Thus, many sites are not as permissive as recommended [20, 31, 37].

Case Study: Required Word Structure and Letter Start. We manually investigated 100 domains requiring a two-word structure as well as domains enforcing letter start, confirming our inference. We did not identify common platforms or frameworks, but many sites used form validation JS libraries (e.g., jQuery Validation, FormCheck.js, Knockout Validation) to enforce a password regex.

4.3.3 Permissive Parameters. Finally, we evaluate the prevalence of permissive parameter values for our sites, as shown in Table 5. Two widely recommended password policies [10, 19, 20, 31, 37] are disallowing users to choose dictionary words and common breached passwords. We observe limited deployment of such password blocking though, as 72% of sites permit dictionary words as passwords and 88% allow breached passwords. Certain password structures are also often discouraged [20], however we detect limited prevention of these patterns as well. Approximately 71% of sites permit sequences, repeating characters, and personal identifiers (e.g., username) in passwords, and 78% allow all-digit passwords. Recent password guidelines [20, 37] also recommend allowing various types of characters. We observe over 30% of sites do not support spaces, Unicode, or emojis in passwords, and about 30% disallow one of the four most popular special characters ("."", "!", "_", and "#").

Case Study: Accepting Popular Passwords. We assess whether sites accept popular passwords using the top four passwords in a password breach dataset [8]. We list these passwords and their acceptance by sites across ranking ranges in Table 6: 39% of sites accepted the top password and nearly half accepted one of the top four passwords. These sites may be vulnerable to password spraying attacks [29, 49] as their policies permit users to choose popular passwords. We note that most restrictive parameters and password blocklisting would disallow such passwords.

	10K	100K	1M
Restrictive Parameters			
Requires 2 Words	4.8	3.9	2.4
Requires No Arbitrary Special	4.3	3.0	1.8
Any 3 of 4 Classes	6.7	6.4	7.4
Any 2 of 4 Classes	14.9	11.0	9.1
Any 2 of 3 General Classes	10.1	10.0	9.3
Starting With a Letter	1.7	2.0	2.9
Permissive Parameters			
Dictionary Words	83.7	80.1	72.0
Sequential Characters	84.1	79.1	71.7
Repeated Characters	82.2	79.8	71.1
Short Digit-only	38.9	66.2	78.0
Long Digit-Only	57.2	69.4	78.2
Personal Identifier	84.6	78.9	71.4
Space	75.5	73.3	69.0
Unicode	69.7	71.3	67.7
Emoji	59.6	65.8	64.4
Breach Password	84.1	84.8	88.2
1st Popular Special = .	82.7	78.5	70.0
2nd Popular Special = !	83.7	77.7	69.6
3rd Popular Special = _	84.1	78.3	69.7
4th Popular Special = #	82.2	76.4	69.4

Table 5: Policy parameter values for all sites within the Tranco Top 10K, 100K, and Top 1M (full population). For both restrictive and permissive parameters, we list the percent of sites where the parameter value is *True*.

Password	Rank	10K	100K	1M
123456	1	21	39	39
123456789	2	26	46	40
qwerty	3	22	38	42
password	4	27	49	48
	Top 4	27	51	53

Table 6: Percentages of signup sites accepting the top four most popular passwords (based on a breach dataset [8]).

4.4 Adherence to Standards and Guidelines

Over time, various organizations have released password policy guidelines. Here, we assess the extent to which sites adhere to these guidelines. In Table 7, we list 9 prominent guidelines in order of publication year, including different security levels offered by some. Appendix Table 9 summarizes these recommendations. While we can determine if a site's policy adheres to a standard, we do not know if the site's owners explicitly chose to follow the standard.

We observe that NIST's 2004 guidelines have been most widely adopted, with 42.1% of sites adhering. Meanwhile, 30.8% of sites' policies satisfy NIST 2017's guidelines, although 16.7% of sites exhibit policies that follow NIST's old 1985 recommendation. These results indicate the staying power of recommendations, as old NIST guidelines are still observed on most sites, even more than 5 years after updated guidelines were released. Similarly, fewer websites adhere to Germany BSI's latest guidelines compared to older ones.

Across NIST and DISA guidelines, we also observe that stronger security levels are significantly less adopted. For example, only 5.5% of sites have policies satisfying NIST 2004 Level 2, compared to 42.1% for Level 1. We also see low adoption of stricter password guidelines, such as those of US CERT, NCSC, and OWASP. Notably, these guidelines and higher security levels generally required stricter length requirements (particularly $L_{min} = 8$), and checks against dictionary words and breached passwords. This suggests incentives to adopt stronger policies are ineffective and the costs of deploying these strong policy parameters are non-trivial.

4.5 Variation by Website Rankings

Here we consider how password policies differ across websites ranked within the Tranco Top 10K, 100K, and 1M.

Standard Name	1M	100K	10K
NIST 1985 (Low)	16.7	22.1	27.4
NIST 1985 (Med)	7.6	12.8	15.4
NIST 1985 (High)	3.8	7.4	9.1
NIST 2004 (Lvl 1)	42.1	65.3	77.9
NIST 2004 (Lvl 2)	5.5	7.5	6.7
BSI 2005	4.8	6.7	6.7
US CERT 2009	0.3	0.4	0.5
DISA 2014 (Med)	4.7	8.8	10.1
DISA 2014 (High)	0.1	0.1	0.5
NIST 2017 (Should)	30.8	40.8	34.6
NIST 2017 (Shall)	1.8	3.16	3.9
NCSC 2018	0.7	1.2	2.4
BSI 2019	14.6	22.3	32.7
BSI 2020	5.9	7.5	7.2
OWASP	1.3	2.9	4.3

Table 7: Percent of sites satisfying different guidelines, across the Tranco Top 10K, 100K, and 1M (full population).

Length. Figure 2 shows the CDFs of minimum and maximum passwords lengths, respectively, for all three groups. We observe that in all graphs, the CDFs for top-ranked sites skew towards longer lengths, which is recommended for stronger passwords. The median minimum password length for top 10K sites is 8 characters, compared to 5 and 6 characters for the top 100K and all sites, respectively. Similarly, while about 40% of all sites allow long passwords that are at least 128 characters, 50% and 55% of top 100K and top 10K sites do, respectively (although a higher portion of top-ranked sites cap passwords at 20 or fewer characters than among all sites).

Restrictive and Permissive Parameters. Table 5 depicts the parameter values for all three ranking ranges, showing the percent of sites within each population where a parameter value is true. We observe that overall, top sites are more likely to enforce restrictions on the password (e.g., R_{cmb24} is true for 15% of top 10K sites, compared to 9% of all sites). Top sites are generally more permissive in which special characters they accept, including periods, exclamation marks, underscores, pound signs, space, and Unicode characters (although slightly fewer top sites accept emojis compared to all sites). Surprisingly, top sites also are more permissive of oft-discouraged password patterns, including dictionary words, sequential and repeated characters, and the inclusion of personal identifiers. However, top sites are significantly less likely to accept all-digit passwords, accepted by only 39-57% of top 10K sites compared to 78% of all sites. Top sites are also slightly less likely to allow breached passwords compared to all websites though (84% of the top 10K versus 88% for all). Overall, top sites apply more password composition requirements but also permit more characters/structures (except all-digit passwords).

Adherence to Guidelines. Table 7 lists the adherence to common guidelines across ranking ranges. We observe that across all guidelines, higher-ranked sites generally exhibit higher adherence, suggesting that they are more likely to follow recommendations. However, the most recent guidelines are still only adopted by a minority of sites across all three ranking ranges (see Section 4.4).

5 COMPARISON WITH PRIOR FINDINGS

Prior works on assessing website password policies are small-scale and largely dated [7, 13, 15, 16, 23, 27, 35] (see Section 2). Here, we compare our results with prior findings, to understand how policies may have changed over time, and the insights afforded by a large-scale perspective.

Top Policies and Parameter Values. Prior work assessed policy parameter values, rather than top policies, likely due to small sample sizes. In comparison, our large-scale study identified the top policies, most of which enforced only length constraints, as well as a long tail of policies which are mostly unique to a site.

Length: A recent 2022 analysis of 120 top English sites observed that a minimum length of 8 was most frequently enforced, followed by lengths 6 and 5 [24]. We observe the same for our top 10K sites, with 40% of sites requiring length 8 passwords, 30% requiring length 6, and 7% requiring length 5. However, when considering the top 1M sites, length 5 was the most prevalent, on nearly 40% of sites. Meanwhile, length 6 and length 8 passwords were required by approximately 15% of sites each. Further, [7, 28] observed few sites without length requirements, but at scale, we observed this policy at nearly a quarter of the sites. Thus, our large-scale measurement identified shorter password length minimums on most sites than reported by recent studies focused on top sites.

Prior work observed widespread use of length caps (note, [24] did not investigate length maximums). Seitz et al. [39] observed an average max length of 43 characters, and Wang et al. [49] did not observe any max lengths greater than 64. In contrast, we observe over a third of all sites allowing 128+ character passwords, with a median length cap of 86 (with even fewer sites using length caps among top-ranked sites). As these prior studies are over a half decade ago and of limited scale, it seems likely that sites today have broadly shifted towards accepting longer passwords.

Restrictive and Permissive Parameters: Few works systematically characterized restrictive and permissive parameters, with most highlighting case studies rather than comprehensive analysis. However, prior work [7, 24, 49] observed between 30-50% of sites enforced several restrictive parameters. We observe a smaller fraction, with only 1.8-9.3% of sites employing any given restrictive parameter, although top-ranked sites employed restrictive parameters more. Thus when considering websites at scale, restrictive parameters are less prevalent overall. Earlier work from 2010 [7] also found few sites performing dictionary checks. However, we observed a modest rate today, at 28% ([24] observed 41% on top English sites).

Adherence to Standards and Guidelines. Prior work mostly predates modern password guidelines [7, 13, 15, 16, 23, 27, 35] (e.g., NIST 2017, BIS 2019), and did not identify comprehensive comparisons of password policies with the standards prevalent at a study's publication.

Variation by Website Ranking. Prior work [13, 28] looked at several US university websites, and found that top-ranked sites had weaker policies than lower-ranked ones, although policies were evaluated using an entropy metric with notable limitations [20, 51]. In contrast, our site population is orders of magnitude larger and has substantially broader ranking coverage, and we observe stronger policy characteristics for top sites (e.g., longer length requirements, broader adherence to modern recommendations).

6 CONCLUDING DISCUSSION

In this study, we conducted the largest evaluation of website password creation policies to date, assessing over 20K sites (~135x more sites than prior work). Our results revealed the state of modern web authentication, and identified insecure policies deployed (especially outside of the top sites). Of note, we observed that 75% of sites allow shorter passwords than the recommended 8 characters [10, 19, 20, 31, 37] (with 12% allowing single-character passwords) and 40% cap password lengths below the 64 characters recommendation [10, 20, 37]. Meanwhile, 15% of sites enforce character constraints, which is no longer recommended [20, 31, 37]. Only 12%-28% of sites employ password blocking, as widely advocated [10, 19, 20, 31, 37]. Finally, a third of sites did not support certain password characters as suggested [20, 37], including whitespaces needed for passphrases. Ultimately, only a minority of sites adhered to modern guidelines overall. Here, we synthesize our findings into lessons for moving web authentication forward.

Improving Software Defaults and Implementation Support. Our case studies in Section 4.3 identified that insecure password policy decisions were closely aligned with the default configurations of popular web software (such as WooCommerce and Shopify). These findings demonstrate the influence of software defaults on web authentication, but also illuminate a potential remediation path: if popular web software implemented recommended password policy configurations by default, many websites could be moved to stronger password policies. For example, *nearly half* of our sites with password length minimums below the 8 characters recommended [10, 19, 20, 31, 37] use the Shopify platform and its default 5 characters minimum. Thus, if Shopify increases its default length to 8 characters, potentially a third of our sites would become newly aligned with modern guidelines. We are currently in the process of communicating with platforms identified offering weak default configurations to encourage such changes.

Related to defaults are the feature support by popular web software. We observed in Section 4.3.3 that only a minority of sites blocked passwords with certain characteristics, which is widely recommended [10, 19, 20, 31, 37]. We hypothesize that this arises partly because many popular web platforms do not provide full support for such blocking, so web developers would need to custom implement such functionality. For example, both Python's Django library [5] and the WordPress CMS [6] by default do not support all password checks. By implementing such features (and enabling by default) for popular web frameworks (many of which are open-source), our community can meaningfully improve web authentication.

Promoting Modern Password Guideline Adoption. Our analysis in Section 4.4 revealed that many sites exhibit policies satisfying password guidelines, but primarily more dated versions. This result provides evidence that password guidelines do generally inform the policy decisions of many websites. However, there must be barriers inhibiting the adoption of more recent recommendations.

A lack of awareness may be one barrier. Here, education and outreach efforts can help inform websites about current guidelines. Prior work on web administrator notifications [25, 26, 44, 45] demonstrated that such outreach efforts can drive the remediation of security issues at scale. Future work can also investigate the resources available about web authentication, and identify information sources that should be updated with current recommendations.

In addition, in Section 4.4, we saw different guidelines from various organizations, with sometimes conflicting recommendations. For example, NIST 2017 [20] and OWASP [37] guidelines avoid password complexity requirements, unlike BSI 2020 [19]. A unified password guideline would provide more consistent and clear recommendations to web administrators around the world. We also uncovered that some guidelines (e.g., OWASP, NCSC 2018) are rarely adopted, suggesting that these guidelines are overly strict or lack visibility and incentives to drive adoption.

Even if adopting a new policy, a remaining challenge is the policy update process. How should websites handle passwords created under the old policy? If old passwords are left as is, the new policy's benefits are not realized. Meanwhile, forced password resets are often onerous to users (as seen with the password resets during data breaches). Future work should investigate effective processes for upgrading password creation policies, and integrate them into existing web software. Organizations releasing password guidelines also must be cognizant of the high burden imposed upon websites when adopting new policies, and guidelines must be released with care (e.g., BSI released two guidelines only one year apart [18, 19]).

Standardizing Password Creation Policies to Promote Usability. In Section 4.2, we observed that websites exhibit wildly diverse policies, with many policies unique to one site. This heterogeneity is likely a usability burden during password creation, where users do not know what constraints are enforced on chosen passwords across different sites. This is especially true as we found that few sites explicitly document their password policies (from Section 3.9). Standardizing password policies would significantly reduce this user friction, providing a unified policy across the web.

Such standardization would benefit password managers as well, as many password managers assist users by automatically generating random and strong passwords. To do so correctly, they must generate a password valid under a site's policy, which is inhibited by the diversity of real-world site policies. For example, some sites disallow long passwords or require certain character compositions (from Section 4), which may not be satisfied by a password manager's randomly generated password. We note that even with the absence of standardization, our results help inform password managers of the common policy constraints enforced by most sites. For example, we found that passwords of length 12–16 are the most likely to be accepted, permitted by 96-98% of sites. Our measurement dataset can also be inputted directly to password managers to provide the specific constraints on the sites that we analyzed.

Future Research Directions. Our study highlights avenues for future investigation. One direction is in improving upon our measurement techniques. While our collected dataset is significantly larger than those of prior work [7, 13, 15, 16, 23, 24, 27, 28, 35, 39, 49], we still successfully analyzed only a minority of sites with account signups. Expanding measurement coverage would allow for more generalizable findings and more extensive analysis of authentication policies across different site characteristics. Similarly, longitudinal measurements could afford insights into policy evolution. Future work could also investigate which website characteristics

[5] https://docs.djangoproject.com/en/4.2/topics/auth/passwords/
[6] https://www.wpbeginner.com/plugins/how-to-force-strong-password-on-users-in-wordpress

correlate with secure and usable password policies, such as website categories, geographic regions, and languages.

7 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. The first author was supported by the Kuwait University Scholarship. This work was also supported in part by the National Science Foundation award CNS-2055549. The opinions expressed in this paper do not necessarily reflect those of the research sponsors.

REFERENCES

[1] 2020. SecLists / Passwords /Common-Credentials. https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials
[2] 2023. *Auto Captcha Solver Service and Cheap Captcha Bypass Service Provider - AZcaptchas.* https://azcaptcha.com/
[3] 2023. *The Proxy API For Web Scraping.* https://www.scraperapi.com/
[4] 2023. *Selenium.* https://www.selenium.dev/
[5] 2023. *Welcome to Faker's documentation!* https://faker.readthedocs.io/
[6] FIDO (Fast IDentity Online) Alliance. 2022. *FIDO Alliance study reveals global password usage is down - yet its continued dominance is proving costly.* https://fidoalliance.org/barometer-2022/
[7] Joseph Bonneau and Sören Preibusch. 2010. The Password Thicket: Technical and Market Failures in Human Authentication on the Web.. In *Workshop on the Economics of Information Security (WEIS).*
[8] Mark Burnett. 2015. *Today I Am Releasing Ten Million Passwords.* https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495
[9] William Burr, Donna Dodson, and W. Timothy Polk. 2014. Electronic Authentication Guidelines. *NIST Special Publication* 800 (2014), 63–1.
[10] Cybersecurity and Infrastructure Security Agency (CISA). 2009. Security Tip (ST04-002) Choosing and Protecting Passwords. (2009).
[11] Joe DeBlasio, Stefan Savage, Geoffrey M Voelker, and Alex C Snoeren. 2017. Tripwire: Inferring internet site compromise. In *ACM Internet Measurement Conference (IMC).*
[12] Kostas Drakonakis, Sotiris Ioannidis, and Jason Polakis. 2020. The cookie hunter: Automated black-box auditing for web authentication and authorization flaws. In *ACM Conference on Computer and Communications Security (CCS).*
[13] Dinei Florêncio and Cormac Herley. 2010. Where do security policies come from?. In *Symposium on Usable Privacy and Security (SOUPS).*
[14] Defense Information Systems Agency (DISA) for United States Department of Defense (DoD). 2014. Application Security and Development Security Technical Implementation Guide. (2014).
[15] Steven Furnell. 2007. An assessment of website password practices. *Computers & Security* 26, 7-8 (2007), 445–451.
[16] Steven Furnell. 2011. Assessing password guidance and enforcement on leading websites. *Computer Fraud & Security* 2011, 12 (2011), 10–18.
[17] Bundesamt für Sicherheit in der Informationstechnik (BSI). 2005. BSI-Standard 100-2. IT-Grundschutz Methodology. (2005).
[18] Bundesamt für Sicherheit in der Informationstechnik (BSI). 2019. IT-Grundschutz Compendium. (2019).
[19] Bundesamt für Sicherheit in der Informationstechnik (BSI). 2020. IT-Grundschutz-Kompendium. (2020).
[20] P Grassi, Michael E Garcia, and James L Fenton. 2017. Digital identity guidelines. *NIST Special Publication* 800 (2017), 63–3.
[21] Philip G Inglesant and M Angela Sasse. 2010. The true cost of unusable password policies: password use in the wild. In *SIGCHI Conference on Human Factors in Computing Systems (CHI).*
[22] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. 2011. Of passwords and people: measuring the effect of password-composition policies. In *SIGCHI Conference on Human Factors in Computing Systems (CHI).*
[23] Bryan Thomas Kuhn and Chlotia Garrison. 2009. A survey of passwords from 2007 to 2009. In *Information Security Curriculum Development Conference.*
[24] Kevin Lee, Sten Sjöberg, and Arvind Narayanan. 2022. Password policies of most top websites fail to follow best practices. In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022).* USENIX Association, Boston, MA, 561–580. https://www.usenix.org/conference/soups2022/presentation/lee
[25] Frank Li, Zakir Durumeric, Jakub Czyz, Mohammad Karami, Michael Bailey, Damon McCoy, Stefan Savage, and Vern Paxson. 2016. You've Got Vulnerability: Exploring Effective Vulnerability Notifications. In *USENIX Security Symposium.*
[26] Frank Li, Grant Ho, Eric Kuan, Yuan Niu, Lucas Ballard, Kurt Thomas, Elie Bursztein, and Vern Paxson. 2016. Remedying Web Hijacking: Notification Effectiveness and Webmaster Comprehension. In *International World Wide Web Conference (WWW).*
[27] Mohammad Mannan and Paul C Van Oorschot. 2008. Security and usability: the gap in real-world online banking. In *Workshop on New Security Paradigms.*
[28] Peter Mayer, Jan Kirchner, and Melanie Volkamer. 2017. A second look at password composition policies in the wild: Comparing samples from 2010 and 2016. In *Symposium on Usable Privacy and Security (SOUPS).*
[29] Mitre. 2023. *Brute Force: Password Spraying.* https://attack.mitre.org/techniques/T1110/003/
[30] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2010. Re: CAPTCHAs—Understanding CAPTCHA-Solving Services in an Economic Context. In *USENIX Security Symposium.*
[31] National Cyber Security Centre (NCSC). 2018. Password administration for system owners. *NIST Special Publication* (2018).
[32] Peter Norvig. 2012. *English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU.* http://norvig.com/mayzner.html
[33] National Bureau of Standards. 1985. "Password Usage" Guidelines. (1985).
[34] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Network and Distributed Systems Security Symposium (NDSS).*
[35] Sören Preibusch and Joseph Bonneau. 2010. The password game: Negative externalities from weak password practices. In *International Conference on Decision and Game Theory for Security.*
[36] Oxford University Press. 2011. *The Oxford English Corpus: Facts about the language.* https://web.archive.org/web/20111226085859/http:/oxforddictionaries.com/words/the-oec-facts-about-the-language
[37] The Open Web Application Security Project. 2023. (2023). https://cheatsheetseries.owasp.org/
[38] Scikit-learn. 2023. *sklearn.svm.SVC'.* https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[39] Tobias Seitz, Manuel Hartmann, Jakob Pfab, and Samuel Souque. 2017. Do differences in password policies prevent password reuse?. In *SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems.*
[40] Richard Shay, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Alain Forget, Saranga Komanduri, Michelle L Mazurek, William Melicher, Sean M Segreti, and Blase Ur. 2015. A spoonful of sugar? The impact of guidance and feedback on password-creation behavior. In *SIGCHI Conference on Human Factors in Computing Systems (CHI).*
[41] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2014. Can long passwords be secure and usable?. In *SIGCHI Conference on Human Factors in Computing Systems (CHI).*
[42] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Designing password policies for strength and usability. *ACM Transactions on Information and System Security (TISSEC)* 18, 4 (2016), 1–34.
[43] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2010. Encountering stronger password requirements: user attitudes and behaviors. In *Symposium on Usable Privacy and Security (SOUPS).*
[44] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. 2018. Didn't You Hear Me? Towards More Successful Web Vulnerability Notifications. In *Network and Distributed System Security Symposium (NDSS).*
[45] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. 2016. Hey, You Have a Problem: On the Feasibility of Large-Scale Web Vulnerability Notification. In *USENIX Security Symposium.*
[46] Blase Ur, Jonathan Bees, Sean M Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Do Users' Perceptions of Password Security Match Reality?. In *ACM CHI Conference on Human Factors in Computing Systems (CHI).*
[47] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, et al. 2012. How does your password measure up? The effect of strength meters on password creation. In *USENIX Security Symposium.*
[48] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2015. I Added '!' at the End to Make It Secure": Observing Password Creation in the Lab. In *Symposium on Usable Privacy and Security (SOUPS).*
[49] Ding Wang and Ping Wang. 2015. The emperor's new password creation policies. In *European Symposium on Research in Computer Security (ESORICS).*
[50] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. 2016. Understanding password choices: How frequently entered passwords are re-used across websites. In *Symposium on Usable Privacy and Security (SOUPS).*
[51] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *ACM Conference on Computer and Communications Security (CCS).*
[52] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *USENIX Security Symposium.*

A APPENDIX

An extended version (EV) of this paper with additional appendices is at https://arxiv.org/abs/2309.03384.

Paper	Year	# Websites
S. Furnell [15]	2007	10
Mannan and Van Oorschot [27]	2007	5
Kuhn et al.[23]	2009	69
Florencio et al. [13]	2010	75
Bonneau et al. [7]	2010	150
Preibusch and Bonneau [35]	2010	150
S. Furnell [16]	2011	10
Wang et al. [49]	2015	50
Seitz et al. [39]	2017	83
Mayer et al. [28]	2017	137
Lee et al. [24]	2022	120

Table 8: Summary of prior work investigating real-world website password policies. We list the paper with its publication year and the number of websites analyzed.

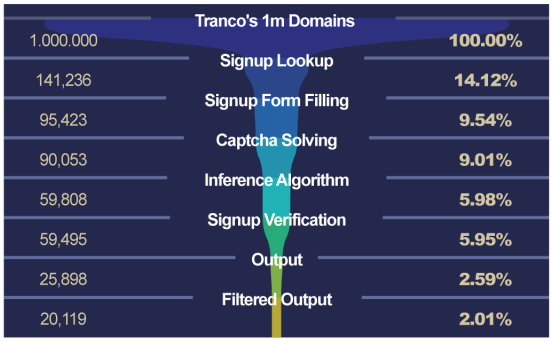


Figure 3: Funnel chart of the one million domains as they flow through the main stages of the framework.

Reference	Year	Country	Guidelines and Recommendations
NIST [33]	1985	USA	**Low Security Level:** Min Length>=4, Requires Digits **Medium:** Min Length>=4, Requires Uppercase, Lowercase, and Digits **High:** Min Length>=6, Requires Uppercase, Lowercase, Digits, and Symbol
NIST [9]	2004	USA	**Level 1:** Min Length>=6, Allows Special Character **Level 2:** Min Length>=8, Allows Special Character, Dictionary (Common) Check, Has composition rules
BSI [17]	2005	DE	Min Length>=8, Requires a digit or a symbol, dictionary (Common) Check
US-CERT [10]	2009	USA	Min Length>=8, Max Length>=64, Dictionary (words) Check, No Personal Information, Requires Uppercase, Lowercase, Digits, and Symbol
DISA [14]	2014	USA	**High Severity:** Min Length>=15 **Medium Severity:** Requires Uppercase, Lowercase, Digits, and Symbol
NIST [20]	2017	USA	**Shall:** Min Length>=8, Dictionary (Breach) Check, Dictionary (Words) Check, No Repetitive, No Sequential **Should:** Max Length>=64, Accepts Space, all Printable ASCII, Unicode including Emoji, No other Composition Rules
NCSC [31]	2018	UK	Dictionary (Common) Check, No Complexity Requirements, No Short Passwords*
BSI [18]	2019	DE	Minimal length = Sufficient*, Complexity = Sufficient*
BSI [19]	2020	DE	Minimal length = Sufficient*, Complexity = Sufficient*, Dictionary (Common) Check
OWASP [37]	2022	-	Min length>=8, Max length>=64, Allow Unicode, Allow Space, Dictionary (Breach) Check

Table 9: Summary of the password policy recommendations provided by multiple organizations. *We interpreted short length by NCSC to be less than 8 characters, sufficient length by BSI 2019 and BSI 2020 to be at least 8 characters long, and sufficient complexity to have at least 2 character classes present.