A Large-Scale Measurement of Website Login Policies

Suood Al Roomi
Georgia Institute of Technology
Kuwait University

Frank Li
Georgia Institute of Technology

Abstract

Authenticating on a website using a password involves a multi-stage login process, where each stage entails critical policy and implementation decisions that impact login security and usability. While the security community has identified best practices for each stage of the login workflow, we currently lack a broad understanding of website login policies in practice. Prior work relied upon manual inspection of websites, producing evaluations of only a small population of sites skewed towards the most popular ones.

In this work, we seek to provide a more comprehensive and systematic picture of real-world website login policies. We develop an automated method for inferring website login policies and apply it to domains across the Google CrUX Top 1 Million. We successfully evaluate the login policies on between 18K and 359K sites (varying depending on the login stage considered), providing characterization of a population two to three orders of magnitude larger than previous studies. Our findings reveal the extent to which insecure login policies exist and identify some underlying causes. Ultimately, our study provides the most comprehensive empirical grounding to date on the state of website login security, shedding light on directions for improving online authentication.

1 Introduction

Passwords remain the dominant mechanism for online account authentication today. While logging into a website using a password may appear like a simple process, it actually involves a multi-stage workflow where account security and privacy are impacted by the website's authentication policy and implementation decisions. In particular, a website must make various design decisions on managing password entry, submission, storage, and validation, as well as handling login failures (e.g., login failure messages and rate limiting).

Over the years, the security community has identified best practices for promoting security and usability at each stage of the login workflow (e.g., [1–3]). Ultimately though, websites must implement these recommendations to improve online authentication in practice. To date, we lack a large-scale understanding of the authentication policies that websites actually employ during the login workflow. Prior work (e.g., [4–7]) has relied upon manual investigation to evaluate website login policies, resulting in a limited characterization that is skewed heavily towards top sites. It is vital for the security community to develop a more comprehensive and systematic understanding of how websites manage password logins, to identify shortcomings in modern online authentication and determine directions for driving better authentication.

In this work, we seek to fill this gap through a large-scale measurement of website login policies, evaluating the policies and implementation decisions at each stage of the login workflow. Doing so is challenging given the extraordinary diversity of the web and that login policy decisions are rarely explicitly published. However, through systematically investigating real-world websites and their login workflows, we develop a web measurement technique for automatically inferring the website login policies in a blackbox fashion. Our method entails automatically creating test accounts on websites, and systematically assessing each login stage to determine the authentication policies enacted. We apply our technique across domains in the Google CrUX [8] Top 1 Million, successfully inferring login policies on between 18K and 359K websites (varying depending on the login stage considered), which is two to three *orders of magnitude* more sites than prior studies.

This large-scale measurement reveals several key findings:

- While HTTPS has been broadly deployed for login workflows, we detect a sizable population of sites still serve login pages and transmit account credentials unencrypted, including government and educational domains.
- We identify sites disallowing copy-pasting of login credentials and storing passwords in plaintext, which are both discouraged by modern recommendations [1, 3, 9].
- Previously, Facebook was the primary documented example of a site deploying typo-tolerant password authentication, where passwords with common typographical errors are still accepted. We find hundreds of sites with typo-tolerant password authentication, including a top 50 site (Pinterest).

- We uncover nearly 6K sites providing login failure messages that enable user enumeration attacks, and identify popular web platforms (e.g., WordPress) driving this issue.
- We find that only a minority of sites employ login rate limiting to prevent online brute-force password guessing attacks, as commonly recommended [1, 2].

Ultimately, our study offers the most expansive survey of modern website login security, establishing empirical grounding on how sites secure their login workflows. Our results highlight the extent to which login security concerns persist today, and reveal avenues for improving online authentication.

2 Website Login Policies

Our study aims to characterize the various authentication policies and implementation decisions made throughout the account login process, which we generically refer to as login policies. To clearly scope our investigation, we first dissect the login workflow into individual stages and discuss the relevant policy and implementation decisions that we investigate.

1. Login Page Visit: First, a user must visit the website's login page. The key implementation decision here is whether this page and its resources are served over HTTPS, which is necessary for providing a secure login environment.

2. Account Credential Entry: Next, a user enters their username and password into the login form. Here, websites can decide whether to allow copy-pasting into these form fields. Modern guidelines [1, 3, 9, 10] recommend permitting such copy-pasting to promote password security (by allowing users to save long complex password to copy over, such as with a password manager) and usability (by avoiding manual password entry, which can result in typographical mistakes).

3. Password Transmission: Upon login form submission, the account credentials are transmitted to the website. As with the login page visit, the crucial implementation decision here is whether this transmission is over a secure TLS channel, otherwise the credentials may be compromised in transit.

4. Password Storage/Retrieval: At the web server, the account's real password is stored in some form, and retrieved for comparison with the user-submitted password. The core implementation decision here is whether the password is stored in plaintext (or some other recoverable format). Plaintext passwords are universally discouraged [1, 2] as a compromise of the credentials database would result in leaked passwords.

5. Password Validation: If the user-provided password matches the stored one, then the login completes successfully. Here, websites may support a typo-tolerant password policy where certain common typographical errors (e.g., missing first or last character, reversed case of letter characters) are permitted, still allowing for login success. Prior analysis of such typo-tolerant schemes [11] demonstrated that typo-tolerance results in improved login usability, with reduced login friction and more successful logins. While the initial security analysis concluded that typo-tolerance results in negligible degradation of authentication security, subsequent analysis [12] found that typo-tolerance's security degradation is significantly larger due to credential stuffing and tweaking attacks. As a consequence, while typo-tolerance does provide notable usability benefits, it also harms password security.

6. Login Failures: If the user-provided credentials do not match a stored one, then the login fails. However, websites can make multiple decisions in handling login failures.

The first decision is in the error message displayed to users. Certain types of messages can leak whether the username/account exists, potentially allowing attackers to enumerate user accounts. Current guidelines advise against revealing specifically whether the submitted username or the password is incorrect, to avoid such user enumeration vulnerabilities [1].

The second decision is in rate limiting or blocking repeated failed logins, to prevent online brute-force password guessing attacks. Existing standards recommend enacting such policies to prevent brute-force online attacks [1, 2].

3 Method

In this section, we describe our method to automatically evaluate the different login policies of websites at scale (illustrated in Figure 1). To design and evaluate our method systematically, we randomly select a set of domains to manually analyze, collecting ground-truth data on website behavior (Section 3.1). We design a process for discovering and completing the account authentication workflows on a domain, which should results in a test account on the site that we can successfully log into (Section 3.2). Then, we devise methods for interrogating each login policy component separately for a domain, through automated tests during the authentication workflows (Section 3.3). In Section 3.4, we describe how we piece together this automation to analyze websites at scale (highlighting limitations and ethical considerations in Sections 3.5 and 3.6, respectively).

3.1 Ground-Truth Analysis

Modern login workflows are diverse in their implementation and design. To develop methods for automatically identifying and classifying relevant webpages and HTML elements, we rely upon heuristics and machine learning models (similar to prior work on automated online account analysis [13, 14]). To ensure that our heuristics and models reflect the characteristics of real-world websites, we randomly sample 2800 domains from across the Tranco Top 1M [15] (using the Dec 13, 2021 snapshot) and manually label domains and relevant HTML elements. We refer to this dataset as the *ground-truth data*. For keyword-based heuristics, we use the standard TF-IDF ranking [16] to determine the most relevant keywords across domains (as detailed in Appendix A). Machine learning models are evaluated and tuned on this ground-truth dataset, with the employed model trained on the full dataset (more details on these models are provided in the following sections). Since each classification task is distinct, a different model may be most suitable. Thus, for each task, we tested

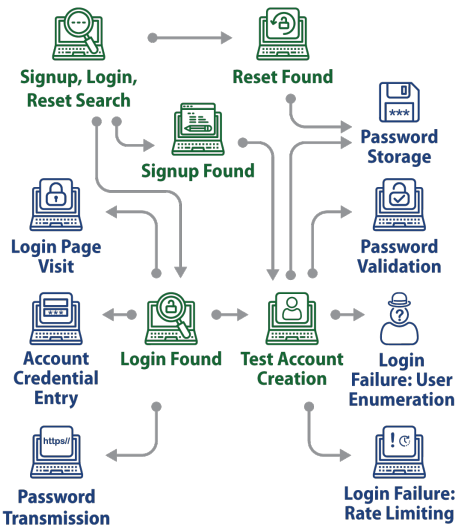

Figure 1: Overview of our automated method for evaluating a domain's login policy. We first search for account signup, login, and password reset pages. If a login page is found, we assess the domain's policy for the login page visit, account credential entry, and password transmission. If a signup page is also found, we create test accounts to assess the domain's handling of password validation and login failures (for both user enumeration and rate limiting aspects). Finally, we infer the domain's password storage policy based on monitoring the test accounts, performing a password reset if found.

different classifiers on our ground-truth data and selected the best-performing one. We note that our data size (2800) balanced the manual work needed to collect the data versus having enough data to train accurate classifiers and heuristics

3.2 Discovering and Completing Account Authentication Workflows

Before evaluating a domain's login policy, we first require identifying its account authentication workflows (if they exist), and proceeding through account creation to generate a test account. Here, we describe each step of this process.

3.2.1 Detecting Account Signup, Login, and Password Reset Forms

Our measurement method centers on automatically creating and logging into a test account, to assess the login workflow. To identify account signup and login HTML forms, we train an SVM binary classifier. The classifier's features include the presence of relevant signup and login-related keywords in the HTML form's title, ID, class, and action (and in any child HTML elements). Other features include the number of password inputs and the total number of form inputs. As training data, we manually label all HTML forms in our ground-truth dataset. We implement our model using Python's sklearn [17], selecting hyper-parameters through grid search and evaluating models with 10-fold cross-validation. Our optimized model exhibits an accuracy of 94.6% (precision = 77.1%, recall = 99.5%) and 94.0% (precision = 94.5%, recall = 96.3%) for classifying account signup and login forms, respectively. (Note, while our signup form classifier exhibited lower precision, its false positives do not propagate into our measurement as we will fail to successfully create an account in such cases.)

We also proceed through the password reset process to identify if passwords may be stored at a website in plaintext. We found from our ground-truth data that we could reliably identify password reset forms through the presence of relevant keywords (selected using TF-IDF) in the HTML form's title, ID, class, and action (and in any child HTML elements), rather than requiring a machine learning model.

3.2.2 Discovering Account Signup, Login, and Password Reset Pages

For a given domain, we search for account signup, login, and password reset pages by looking for pages with the associated HTML forms (detected as discussed in Section 3.2.1). Specifically, our search process entails the following steps, stopping once pages with the forms are discovered:

1. We first start with the landing page and look for account signup, login, and password reset forms.
2. We next crawl candidate URL links on the landing page with common signup, login, and password reset keywords (selected by applying TF-IDF on our ground-truth data, as discussed in Appendix A). On each linked page, we again look for the relevant HTML forms. If the forms are not all found, we do one final crawl of candidate URLs on the pages linked to from the landing page. To avoid excessively crawling a single domain, we limit the number of candidate URLs crawled to four per page. (From our ground-truth data, we observed that this threshold was effective for discovering signup, login, and password reset URLs, as most pages had few, if any, candidate URLs.)
3. Finally, we query the Google search engine for the domain's account signup, login, and password reset pages (using ScraperAPI [18]). The signup search query has the domain name and "account OR register OR sign+up OR create", the login query combines the domain name and "account AND signin", and the password reset query uses the domain name and "reset OR recover OR forgot OR password OR lookup+credentials". These queries were constructed using the most relevant keywords (selected using TF-IDF) in the HTML titles of signup, login, and password reset pages in our ground-truth data. We crawl and inspect candidate URLs (those with relevant keywords in the URL, as done in the previous step) from the search results, evaluating up to four candidate URLs (a threshold observed as sufficient for our ground-truth dataset).

We briefly note that we avoid re-analyzing candidates URLs previously evaluated (e.g., the same links at different parts of a page or on different pages). Also, our crawling method is non-interactive and does not simulate user actions

on pages (e.g., scrolling, clicking on different parts of the page) besides form filling. While we observed some sites requiring interaction for signup or login forms to appear, this behavior was not widespread in our ground-truth data, and automating the interaction is challenging (given the diversity of potential actions involved).

3.2.3 Attempting Account Signup, Login, and Password Reset

With account signup, login, and password reset pages found, we can attempt to fill and submit signup, login, and password reset forms automatically, starting with signup forms to create a test account. Here there are two key challenges.

1. Signup form fields must be identified and filled with acceptable values/action. We classify the form fields based on the HTML input element's name, class, and ID, using relevant keywords identified in our ground-truth data.
   - *Email Field:* We supply an email under our control. We setup and configure our own email server and email domain, so for each evaluated domain, we dynamically generate a brand new email specific for that test account.
   - *Password Field:* We attempt two passwords ("MxT7zcS4k5-@" and "MxT7zct41S"), similar to prior work [13] (we avoided testing additional passwords to limit the account creation attempts.). Based on prior analyses of website password composition policies [4, 19–21], one of these passwords is likely accepted by the vast majority of websites (99% of websites as evaluated by Seitz et al. [21]) as they contain multiple lowercase, uppercase, and digit characters (with one containing multiple special characters), and are 10-12 characters long (surpassing common password length minimums without exceeding common length maximums).
   - *Other Fields:* For other common form fields (e.g., name, address), we either use pre-selected values (not real user data) or the Faker Python library [22] to generate synthetic data. For unrecognized fields, we use Faker to generate a random string as a last resort.
   - *Submit Button:* For multi-button forms (e.g., signup and single sign-on buttons), we identify the signup button using keywords derived from our ground-truth data.
2. CAPTCHAs may be presented prior to or after submitting the signup form. We identified the presence of CAPTCHAs on 49% of signup forms in our ground-truth data. We decided to solve these CAPTCHAs to significantly increase the likelihood of successfully assessing a site. Given the scale of our work and ethical concerns, we decided against using human-driven CAPTCHA solvers and opted for an automated CAPTCHA solver, AZcaptcha [23]. We identified CAPTCHAs in the signup form by fingerprinting the HTML and JavaScript code associated with popular CAPTCHAs handled by AZcaptcha, and pass the extracted CAPTCHAs to AZcaptcha to solve.

Completing login and password reset forms also share these two challenges. We identify the username/email field and the password field (when logging in) using a similar keyword-driven approach, and supply the appropriate values used during the signup process. We use the same method as with signup forms to handle CAPTCHAs.

3.2.4 Determining Account Signup and Login Success

While the process in Section 3.2.3 results in a submitted account signup or login form, our provided data and actions may be incorrect, resulting in failed signups/logins. Websites also are diverse in what happens after submitting these forms (e.g., some redirect to a new page, while others modify the existing page). To avoid incorrectly analyzing sites that we did not successfully submit these forms on, we developed ensemble decision tree classifiers for signup/login success that operates on page features returned upon form submissions.

The signup success classifier was trained using data from manual signup attempts on a randomly selected 160 domains in our ground-truth data. Our features include keywords in the response page and URL, the similarity of the page and its URL before and after form submission, and the presence of the signup form. We trained an XGBoost decision tree ensemble model with 100 trees, tuning hyperparameters using grid search and using 4-fold cross-validation model. Our model exhibited a 91.3% average accuracy during cross-validation (precision = 94.3%, recall = 90.9%).

Similarly, the login success classifier used features such as the presence of the login form after submission, the presence of a page redirection, and keywords in the response page and URL. Using the feature values collected from manual logins on 250 domains[1] in our ground-truth data, we trained an XGboost decision tree ensemble model with 50 trees, again using grid search to select hyperparameters, and evaluating using 4-fold cross validation. Our cross-validation model had an 94% average accuracy (precision = 98.4%, recall = 96.0%).

Note that we do not need to verify whether password resets are successful as subsequent analysis is not dependent on successful password resets (as discussed in Section 3.3.4).

3.2.5 Account Verification

Some websites require users to verify the supplied email address, in order to complete the account signup process and allow logins. In the accounts manually created on domains in our ground-truth data, we observed 39% of domains sending verification emails, all containing a verification URL that once visited, completed the verification process. In all cases, the verification email was the first email received from that domain (expected as a site should not send additional emails to an account before verifying it), and it arrived within 60 minutes of account signup (with all but two domains sending the verification within 20 minutes).

[1]While we trained our signup success classifier first, using a smaller training dataset, we found that we needed to collect more training data to improve the accuracy of the login success classifier.

As mentioned in Section 3.2.3, we set up our own email server and domain, and provide new email addresses when creating a test account. Thus we have full visibility and control over emails received. To complete account verification (if it occurs), we monitor for emails sent by a domain within 60 minutes of account signup, and click the links in the emails. We only verify accounts through email, and do not engage in other verification processes (e.g., phone verification).

3.2.6 Sanity Checks

At this stage of our method, we should now have created an account on a website that we can successfully log into. However, as our automated method relied upon heuristics and classifiers that are not perfectly accurate, we may have introduced errors during the account creation process. False negative cases may have arisen as our method did not successfully identify and proceed through the account creation workflow, resulting in sites missing that we could have later analyzed. False positive cases are arguably more concerning, as we believe we have an account on a site when we do not, and thus may proceed with incorrectly analyzing login policies. To reduce false positives in our dataset, we conduct one final sanity check after account creation. We make two login attempts, one with the correct credentials and another with incorrect ones. We only proceed with evaluating a site if our login success classifier classifies both the correct login attempt as successful and the incorrect one as unsuccessful.

3.3 Login Policy Evaluations

For a given domain, we next evaluate the authentication policies and implementation decisions at each stage of the login workflow (if possible), as discussed in Section 2. Note that for a given domain supporting online accounts, we may not successfully complete all steps in the account authentication workflows (Section 3.2), which prevents evaluating all login policy dimensions. However, partial completion of the steps may still allow us to assess a subset of the policy aspects. For example, if we can identify the login page and form, we can still investigate how the website handles the first three stages of the login workflow (i.e., login page visit, account credential entry, and password transmission), regardless of whether we are able to successfully register an account. When analyzing a specific login policy aspect, we evaluate all domains for which we possibly can, even if we cannot assess the full login policy for some of those domains. This affords the largest-scale measurement for each login policy component.

3.3.1 Login Page Visit

We assess this login policy component for all domains where the login page is found. We crawl the login page using both HTTP and HTTPS, recording any HTTP errors or redirections. For login pages loaded over HTTPS, we monitor the page for 60 seconds while recording the browser error log (in this case, we use the Google Chrome browser), specifically looking for mixed content errors (where some page resources are loaded over HTTP). We categorize two types of mixed content errors (as outputted by Google Chrome), "Warning" where the browser was able to upgrade the resource load to HTTPS, and "Severe" where the browser was not able to upgrade the HTTP resource. We classify a mixed content login page by the most severe warning observed.

3.3.2 Account Credential Entry

For all domains where the login page is found (which also indicates that a login form was identified), we crawl the login page with a full Chrome browser (closing any popups/dialogs that appear, as we observed such activity on sites in our ground-truth dataset). Using Selenium's Actions API [24] for emulating user actions during browser automation, we identify the username/email and password fields in the login form, clear any data in those fields if present, and attempt to copy-paste distinct test values into each field. We then check whether a field's content now contains the test value, indicating that copy-pasting is allowed for that field (note, password masking does not interfere with our assessment as we inspect the content of the password form field itself).

3.3.3 Password Transmission

For domains with identified login forms, we see if the form data is sent over HTTP based on the form's action attribute.

3.3.4 Password Storage/Retrieval

We seek to identify if a website stores/accesses passwords in plaintext. However, as storage is managed server-side, we lack direct visibility. Instead, we attempt to discover sites storing plaintext passwords by monitoring for sites sending the plaintext password to the account email (which is under our control, as discussed in Section 3.2.5), including after a password reset. For all domains where we found the password reset page and detected successful account signup, we initiate the password reset by filling out and submitting the password reset form (as described in Sections 3.2.3).

3.3.5 Password Validation

To assess a domain's typo-tolerance policy, we test logging in using passwords with typos. This evaluation requires domains where we successfully completed the authentication workflow steps (in Section 3.2), and have test accounts we can log into.

Prior work [11, 12] identified five common typos suitable for correction by a typo-tolerant scheme: incorrect letter casing for only the first password character, inadvertent caps lock resulting in inverted letter casing in the entire password, an extraneous character at the password start, an extraneous character at the password end, and a missing shift key applied to the last character. However, we observe that the behavior of caps lock differs between OSes when combined with the shift key (i.e., when typing an uppercase letter with the caps lock on). On Mac OS, the character will remain an uppercase letter, while on Windows, the character will be lowercase (as the shift key reverses the caps lock). Thus, we expand the caps

	MxT7zcS4k5-@	**MxT7zct41S**
1. Incorrect Case (first)	mxT7zcS4k5-@	mxT7zct41S
2. Cap Locks: Inverted Case	mXt7ZCs4K5-@	mXt7ZCT41s
3. Cap Locks: All Uppercase	MXT7ZCS4K5-@	MXT7ZCT41S
4. Extra Character (front)	NMxT7zcS4k5-@	NMxT7zct41S
5. Extra Character (end)	MxT7zcS4k5-@1	MxT7zct41S1
6. Missing Shift Key (last)	MxT7zcS4k5-2	MxT7zct41s

Table 1: We test the six typos permitted by typo-tolerant password policies in prior work [11]. We list the original passwords used in our measurements (bolded), and show examples of each typo on those passwords.

lock typo into two different incorrect passwords, resulting in six total passwords with typos that we consider. Table 1 lists the six typos considered, and what the passwords we used to create accounts would be with each typo.

For each typo, we apply that typo to an account's actual password on a domain (as shown in Table 1), and attempt logging in. Note that when adding an extra character, we only use a letter or a number character, rather than a special symbol character, to avoid violating any password composition constraints. If the login succeeds (as classified by the login success classifier, described in Section 3.2.4), we conclude that the domain employs a typo-tolerant policy, and track which typos are accepted. Note that as this evaluation requires multiple login attempts, we structure our large-scale measurement such that each login attempt is at least an hour apart (in practice, much longer), as described in Section 3.4.

3.3.6 Login Failures: User Enumeration

We test for user enumeration vulnerabilities only on domains where we successfully created an account and could log in. On these domains, we collect login failure messages by attempting two logins with incorrect credentials, one using the correct username/email but the wrong password, and another using a wrong username/email (generated by appending a random string to the start of the real username/email used for the account). We then save the response HTML pages for both failed login attempts to analyze their error messages. (Note, our sanity check in Section 3.2.6 already attempts one login with the correct username/email but the wrong password, so in practice, we generate only one additional login attempt here with an incorrect username/email.)

The login error messages are extracted from the HTML page using keywords found in the ID, Class, and Name attributes (selected from our ground-truth data). The error messages are then normalized by 1) removing any HTML tags, 2) translating non-English messages (as detected by Python's langdetect library [25]) to English (using the Python Translate library [26]), 3) replacing the account email, username, and password strings in the message with entity labels (e.g., "[email]"), and 4) lowercasing all letters.

To classify whether a login error message enables user enumeration, we first manually collect and label the error messages on 250 domains in our ground-truth data. We then used a pre-trained sentence encoder (all-DistilRoBERTa [27]) to generate sentence embeddings for our error messages as input features, and trained a XGBoost decision tree model, again using grid search to select hyperparameters, and evaluating using 5-fold cross-validation. Our cross-validation model exhibited an 88% average accuracy (precision = 63.0%, recall = 93.6%). As our model exhibited high recall but lower precision, our results may overestimate the vulnerable population, serving more as an upper bound. However, our analysis of the top error messages (Section 4.7) correctly found those with user enumeration vulnerabilities, so our conclusions are largely unaffected by the model's inaccuracies.

3.3.7 Login Failures: Rate Limiting

Websites may use various techniques for classifying suspicious login attempts. Attempting to reverse-engineer exactly how they do so is beyond the scope of this work. Instead, we aim to identify which websites *do not* employ login rate limiting, even for clearly inorganic repeated login attempts.

To analyze these domains, we require that we successfully created and could log into an account. We then execute 15 consecutive login attempts using the correct username/email but an incorrect password, waiting only one second between each attempt (all from the same browser on the same host, without clearing browser cookies between login attempts). After all 15 failed login attempts, we then immediately attempt a login using the correct credentials, recording if we are able to successfully log in. If not, we followed up with one final valid login attempt (using the correct credentials) one hour later. We chose 15 logins to balance between triggering rate limiting and inducing load on a domain.

3.4 Large-Scale Measurement Implementation

The method discussed in Sections 3.2 and 3.3 produces the login policy evaluations for a specific domain. For each site, evaluating the login page visit, account credential entry, and password transmission (Stages 1-3 of the login workflow) does not require creating an account on the site. For evaluating the remaining login policy dimensions, we create two test accounts per domain. One account is used to first evaluate password validation and login failure user enumeration issues, then we finally evaluate password storage/retrieval (as the password reset could potentially result in lost access to our account). We then use the other account to evaluate login failure rate limiting, as the rate limiting can also result in lost account access (note, we create this second account from a distinct IP address, to avoid associating our two test accounts).

We conducted our large-scale measurement in December and January 2023, applying this method across the top 1M domains from the November 20, 2023 snapshot of Google's Chrome User Experience Report (CrUX) [8], which notably contains only website domains visited by browser users (rather than non-web domains derived from other datasets,

such as passive DNS)[2]. For web crawling, we use Selenium browser automation [28] with headless Chrome instances[3]. For each page crawled, we allow 30 seconds for the page to fully load, before analyzing or recording (this threshold was based on observing page load times in our ground-truth data).

To minimize the computational load induced on websites and to also reduce the chance that anti-bot measures are enacted on our test accounts, we rate limit our crawling to at most one page load every 30 seconds per domain and distribute our crawling traffic across a set of proxies within our organization's network for IP diversity. We also heavily rate limit login attempts per domain, such that we wait at least one hour between each login attempt, except for the login failure rate limiting experiment (which requires a burst of login attempts). Due to this rate limiting, we highly parallelize our site evaluation method (Sections 3.2 and 3.3) across domains, such that for each step in our method, we round-robin through all domains first before proceeding to the next step. As a result, in practice, the inter-arrival times between page loads and login attempts on each domain is significantly higher than that enforced by our rate limits.

3.5 Limitations

Our study is a best-effort measurement of website login policies. Performing such a measurement at scale required extensive engineering, but given the diversity of website designs and implementations, our method is still imperfect. We do not believe that alternative non-blackbox measurement approaches would be more effective though, as login policy parameters are rarely published by websites, many policy parameters are handled server-side so they cannot be inferred from client-side analysis, and manual investigations of websites or user studies with website operators do not scale (as done in prior work).

Our measurement has false negatives for sites where we failed to detect and successfully complete authentication workflows, including those with complex workflows (e.g., multi-page forms, unique form fields), registration fees, or offline membership. Thus, prevalence amongst our observed populations does not necessarily generalize to all websites with accounts. However, the set of domains found exhibiting insecure login policies serves as a lower bound on the vulnerable population size. Furthermore, our collected dataset is still orders of magnitude larger and more diverse across rankings than prior studies, which we argue serves as a more generalizable empirical grounding.

Our measurement could potentially also generate false positives on sites where our heuristics and machine learning models misclassify successful authentication workflows. However, such false positives would require errors by multiple distinct classifiers, and we additionally apply a sanity check (discussed in Section 3.2.6) to further reduce the chance of false positives. To instill further confidence in our findings, for each login policy dimension, we randomly sampled and manually analyzed at least 20 domains that our method outputted the policy configuration for. Across our entire measurement, we did not observe any false positive results, signally that the rate of false positives in our data should be low.

One unexpected issue we encountered was with automatically solving CAPTCHAs during repeated login attempts, which resulted in errors analyzing certain policy dimensions for some domains (particularly rate limiting and typo-tolerance). We observed that while our automated CAPTCHA-solving workflow was accurate for the CAPTCHAs observed in our ground-truth dataset, on some evaluated domains, it could fail continuously. For other evaluated domains, it could fail probabilistically (e.g., due to randomization with the CAPTCHA appearance and challenge), so the probability of one login attempt failing across multiple attempts was high enough to prevent analyzing certain policy parameters correctly. We opted not to re-measure sites that we erred on as 1) we are not aware of a more reliable way to automate CAPTCHA-solving (without using human solvers, which entail ethical concerns [29]), and repeated measurements are likely to result in similar outcomes, and 2) we did not want to increase our measurement's footprint on domains through generating additional accounts and login attempts (as will be discussed in Section 3.6 on our ethical considerations).

Appendix B provides a detailed manual evaluation of all stages of our measurement on a random sample of domains.

3.6 Ethical Considerations

As our study involves a large web measurement, there are several important ethical considerations. Our method requires automated account creation and login. Prior studies have performed similar automated account creation [13, 14], and we adopt similar measures as these works to minimize risk/harm in our method. We only create up to two test accounts, and test up to 29 login attempts across both accounts (up to 19 logins on the account used for evaluating login failure rate limiting, and 8 logins on the other account). To avoid inducing computational load on websites and triggering anti-bot measures, we heavily rate limit our crawling and login attempts (as discussed in Section 3.4). In particular, we do not visit pages on a domain faster than once every 30 seconds, and login attempts are at least one hour apart, typically much longer (except for the rate limiting experiment). We believe that for websites supporting online accounts, this rate of crawling and

[2]While we started our ground-truth analysis with randomly sampled websites from Tranco [15], our full measurement used CrUX as that list contains purely websites. However, CrUX does not provide fine-grained rankings (only ranking buckets) so we still use Tranco for fine-grained rankings.

[3]While websites may detect and block headless browser crawler, we did not observe higher crawling success when using full browser instances while debugging our method on our ground-truth dataset. We suspect that many sites either do not block crawlers or apply anti-bot techniques that are similarly effective on full browsers. Thus, we avoid using full instances as they require more compute resources and execution time (except for evaluating account credential entry, which required a full browser GUI to enumate user actions on).

login attempts should incur minimal costs, and should not tax even small websites. Furthermore, the research community has established precedence for creating and analyzing a small number of test accounts for measurement purposes (we note that existing measurement studies on authentication policies also created test accounts, but did so manually [4, 6, 30, 31]). As part of our method, we solve CAPTCHAs with an automated solver. We avoid human-driven CAPTCHA solving services due to ethical concerns previously identified [29].

We additionally consulted our organization's general counsel to evaluate the legal risk of our measurement method, as some of our methods may be contrary to a website's terms of service, 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, as there is support from judicial precedence and there is a lack of reasonable damages/harm incurred by websites. Our organization's administration also reviewed and approved this study. Finally, there are no human subject concerns with this study, as no real user data was used or collected, nor were there interactions with any individuals.

4 Results

Here we describe the dataset collected and our analysis of website login policies. We will make our data available to researchers upon request (due to the data's sensitivity).

4.1 Dataset Collected

As shown in Figure 2, crawling the domains in the Google CruX Top 1 Million [8], we found an account login page on 358.9K domains, a signup page on 258.2K domains, and a password reset page on 27.3K domains[4]. After our automated account signup process (which includes the sanity check discussed in Section 3.2.6), we were able to create an initial test account on 45.0K domains[5]. When creating a second account (for assessing rate limiting), we succeeded on only a subset of 37.3K domains. We suspect that for the 17% of domains where we failed to create a second account, the site detected and blocked our subsequent automated signup attempt.

Our evaluated domains are distributed across the rankings (note that Google CruX only ranks domains in buckets [8]). Domains that we found login pages for were evenly distributed throughout the top 1M, with 3,344 sites in the top 10K and 36.9K domains in the top 100K. The distribution of domains that we were able to successfully create a test account on was also spread throughout the top 1M, but more heavily skewed towards lower-ranked domains: 271 domains in the top 10K and 3,416 domains in the top 100K. (Domains with

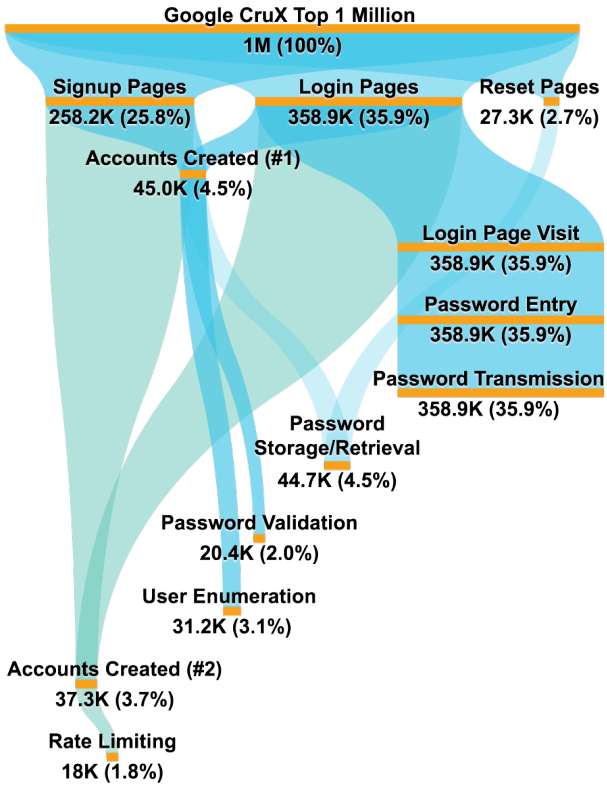

Figure 2: Funnel diagram of our initial 1M domains as they are processed by each stage of our measurement method.

both test accounts created were similar, with 237 domains in the top 10K and 2949 in the top 100K.).

We evaluate the login page visit, password entry, and password transmission policies on all domains where the login form and page were identified. The remaining policy dimensions required domains with test accounts created. In subsequent sections, we analyze each specific policy. (Additionally, Appendix C provides a preliminary characterization of third-party scripts on login pages, which prior work found could exfiltrate login credentials [32].)

4.2 Login Page Visit

As shown in Table 2, across the 358.9K domains with a login page found, we found nearly 2K domains where the login page was served *only* over HTTP, meaning all user logins occur in an insecure environment. We identified an additional 21.2K domains that offered the login page over HTTP, in addition to HTTPS. In such cases, sites should redirect HTTP visitors to HTTPS, lest they authenticate under insecure conditions.

For domains serving login pages over HTTPS, we also looked at whether the login page contained mixed-content served over both HTTP and HTTPS (as HTTP-loaded resources are retrieved over an insecure channel, potentially compromising the secure login environment). Of the 356.9K domains supporting HTTPS login pages, we found 10.3K

[4]Our process for automatically finding password reset workflows was less successful as reset pages reside in deeper parts of websites, may require interaction, may not exist, or may require a different non-form workflow.

[5]Our results exceed other automated account creation works in the literature [13, 14]. For comparison, [14] most recently successfully created accounts on 25K domains out of 1.5M, a 1.7% success rate, compared to our 4.5% success rate.

	HTTP Login Page		HTTPS Login Page Mixed Content		Login Form Submission	
	HTTP Only	HTTP + HTTPS	Warning	Severe	HTTP	Total Analyzed
10K	14	123	28	16	14	3,197
100K	143	1,855	487	370	176	35,458
1M	1967	21,195	5020	5292	2235	358,900

Table 2: The number of domains serving HTTP login pages (HTTP-only and in addition to HTTPS), hosting mixed content HTTPS login pages (only warning-level versus severe mixed content), or submitting login data over HTTP.

domains with mixed content. Of those, half contained only less severe mixed content cases where the browser could automatically upgrade the resource load to use HTTPS, negating mixed content's risk in practice. However, the remaining 5.3K domains exhibited severe mixed content, where some page resources could only be loaded over HTTP.

Differences across Rankings. Across ranking ranges, we observe that the prevalence of insecure login page HTTPS configurations is slightly higher for lower-ranked sites. While 0.41% of top 10K domains had HTTP-only login pages, 0.55% of the top 1M domains were likewise. Meanwhile, 3.7% of the top 10K domains served login pages over both HTTP and HTTPS, compared to 5.9% for top 1M. Mixed content domains shared a similar trend; 1.3% of top 10K domains had mixed content login pages, versus 2.9% for the top 1M.

This observation suggests that lower-ranked sites are more likely to exhibit insecure login page TLS configurations. However, top-ranked sites are not devoid of these security concerns. By manually investigating the 14 domains within the top 10K supporting only HTTP for logins, we found a diverse set of domains, including online forums, adult sites, video streaming platforms, the official government site for a state in India, an electricity utilities payment portal, and a Taiwanese news site.

Case Study: Government and Educational Domains. Among domains serving login pages only over HTTP, we identified two interesting sets of domains based on their SLDs and TLDs: government and educational domains. We found 11 government domains (with *.gov* in the domain SLD or TLD). Based on the country code TLD (ccTLD), 5 of these domains were for India and 3 were for Brazil. We also found 9 educational organizations (with *.edu* in the domain SLD or TLD) serving HTTP-only login pages, including university login portals and educational exam centers.

When expanding to domains supporting HTTP and HTTPS login pages, we found 643 government domains and 820 educational ones. Based on the ccTLD of government domains, the top countries involved were India (204), followed by Brazil (42), Bangladesh (41), and Vietnam (40). For educational domains, many did not include a ccTLD, limiting our country-level association. However, we did observe that the most common ccTLDs were Vietnam (57), Argentina (50), Brazil (46), and Peru (46). Government and educational user accounts are often particularly sensitive, so these sets of domains with insecure login pages are notably problematic (e.g., in the US, student data is specially regulated by FERPA [33]).

	User	Pwd	Both	Either	Total Analyzed
10K	23	42	20	45	3,197
100K	253	341	206	388	35,458
1M	2192	2866	1805	3253	358,900

Table 3: The number of domains disallowing pasting of usernames or passwords into login forms.

4.3 Password Entry

Our results on domains disabling login form copy-pasting is shown in Table 3. Out of 358.9K domains with login pages found, we detected 3.2K of the domains disallowed copy-pasting either the email/username or the password field. While most of these domains (1.8K) disallowed pasting both fields, domains were more likely to disallow pasting passwords than usernames (with 1K domains only disallowing password pasting), presumably exhibiting the now outdated [1] mentality that permitting password pasting is insecure.

Differences across Rankings. Across ranking ranges (as shown in Table 3), we observed similar proportions of analyzed domains disallowing pasting, although top 10K domains disabled pasting slightly more often (1.4%) compared to top 100K domains (1.1%) and top 1M domains (0.9%).

Case Study: Indian Domains. Of the top 10K domains observed disabling pasting, we note that 20% were in India's ccTLD (the most common TLD besides *.com*). Nearly all were government sites (*.gov.in*), hinting that this practice may be broadly adopted amongst Indian government websites.

Case Study: Disabling Pasting. For domains disallowing login form pasting, we analyzed their HTML login forms for the *onpaste* HTML attribute. Only 178 domains (5.6%) used this attribute. Randomly sampling 15 other domains, we identified that they included JavaScript code that created a pasting event listener, preventing the paste event with the preventDefault() method. Thus, most sites use JavaScript for disabling pasting (and that analysis of HTML forms only would mischaracterize pasting allowance for most sites).

	Upon Reg.	After Verif.	After Reset	Total w/ Plaintxt	Total Analyzed
10K	3	1	-	4	263
100K	27	14	2	43	3383
1M	410	134	43	570	44,703

Table 4: The number of domains sending plaintext passwords in emails either upon registration, after email verification, or after password reset.

4.4 Password Transmission

Analyzing the 358.9K domains with login pages found, we uncovered 2.2K domains that could transmit the password over HTTP, as shown in Table 2. These domains are primarily those with HTTP-only login pages; over 96% of such domains (1894) submitted login form data over HTTP. We did find 73 HTTP-only login domains sending credentials over HTTPS though, demonstrating partial HTTPS deployment but not for securing the login page itself. For the remaining 341 domains, we identified that they were domains hosting login pages on both HTTP and HTTPS, and that the login form data was sent to a relative URL. Thus, accessing the login page over HTTP would result in the password also being transmitted in the clear, highlighting the danger of HTTP login page access.

Differences across rankings. As with HTTP-only login pages, we see a slight increase in HTTP password submission for lower-ranked: 0.44% of top 10K domains transmitted passwords in the clear, compared to 0.50% of top 100K domains and 0.62% of top 1M domains.

Case Study: HTTPS Password Transmission on HTTP Login Pages. Manually inspecting a random sample of the HTTP-only login domains sending login credentials over HTTPS, we observed that these domains were typically part of a larger online service, and sent the login credentials to an HTTPS endpoint of the service. For example, several domains were for departments within universities or local governments (similar with Section 4.2) that, while serving the login page over HTTP, sent passwords to a central organization authentication URL. Thus, we hypothesize that in many cases, these are domains for organizations that operate in a distributed fashion, such that while individual sub-organizations configure their local login page (e.g., using only HTTP), the authentication itself is centrally managed (e.g., using HTTPS).

4.5 Password Storage/Retrieval

Recall that on the 45K domains where we successfully created an account, we analyzed all email communications with the account email, monitoring for our account password in emails. We were able to automatically proceed through the password reset on 27.3K of these domains as well (for others, we were unable to automatically locate the password reset workflows, although we also observed that some sites do not seem to publicly support this functionality). In total, we observed 570 domains that sent an email containing our plaintext password, as shown in in Table 4. Appendix Figure 4 shows examples of emails that we received with plaintext passwords. Note that as insecure password storage is a server-side implementation, we lack visibility into a site's configuration unless it sends the plaintext password via email. Thus, our results serve as a lower bound on insecure password storage.

For most of these domains (410, or 72%), one such email came immediately upon account signup, before any action on the account's part (such as email verification). Upon manual inspection, we observe that these are primarily welcome emails providing the account login details. While it is possible that these domains store the password securely, and only transmitted the plaintext password during account creation, this practice is insecure as 1) email is often sent across the Internet over unencrypted channels [34], 2) user inboxes now contain the password in the clear, potentially accessible to email providers or anyone else who obtains access to the emails (e.g., email compromise).

For 134 domains, we received an email with a plaintext password after email verification. Unlike in the above case where the password was immediately sent upon account registration, we believe here that there is stronger evidence of plaintext password storage, as we performed account verification at least an hour after account signup (as mentioned in Section 3.2.5). Finally, for 43 domains, we received the plaintext password via password resets, which provides clear evidence of plaintext password storage. (For 17 such domains, we also received the plaintext password in a prior email.)

Differences across Rankings. Across the ranking ranges, we observed similar percentages of analyzed domains sending emails with plaintext passwords (between 1.27–1.45%).

Case Study: European Websites. Of the 570 domains found sending plaintext passwords in emails, we note that a large portion (147, or 26%) were domains with a ccTLD for a country in the European Union (35 domains in Bulgaria, 18 in Italy, 14 in Poland, and 12 in both France and Germany). The storage of plaintext password by these sites may potentially violate the EU's GDPR Article 32 [35], which requires that European websites securely encrypt user data. In 2018, a German website was fined under GDPR for a data breach while storing plaintext passwords [36]. (We did not observe a large portion of EU domains for other GDPR-related concerns, such as with cleartext password submission.)

Case Study: Comparison with the Plain Text Offender's List. The Plain Text Offenders (PTO) project [37] is a crowdsourced compilation of domains found sending plaintext passwords through emails, collected since 2011 (thus many listed domains may no longer store passwords in plaintext). As of February 2023, the list has 5.8K domains, 2.2K of which are in our top 1M dataset. Of these 2.2K domains, we created an account on 933 domains, and did password resets on 927.

We found 26 of these overlapping domains still sending their passwords in plaintext: 17 did so upon registration, while most of the remaining domains (7) did so after a password reset. Several domains were added to the PTO list in 2011,

Typos Permitted	10K	100K	1M
1. Incorrect Case (first)	5	36	264
2. Cap Locks: Inverted Case	5	36	264
3. Cap Locks: All Uppercase	4	35	251
4. Extra Character (front)	0	0	0
5. Extra Character (end)	1	1	14
6. Missing Shift Key (last)	0	2	20
Fully Case-Insensitive	4	35	251
Any Typo-Tolerance	5	36	273
Total Analyzed	103	1553	20,357

Table 5: The number of domains exhibiting typo-tolerant password authentication, for different typo-tolerant policies.

suggesting that they have stored plaintext passwords for over a decade. We also randomly sampled 10 domains on the PTO list that we analyzed but did not observe emails with plaintext passwords. We manually registered and verified accounts on these sites and conducted password resets. Across the sites, we received welcome, verification, and password reset emails, but none contained our account password. This result indicates that the PTO list has outdated data and that our measurement did not exhibit extensive false positives. In addition, our measurement identified 544 sites not in the PTO list.

4.6 Password Validation

We were able to fully analyze the typo-tolerance policy of 20.4K domains (103 in the top 10K, and 1553 in the top 100K). On other domains, we encountered CAPTCHA-solving errors after the repeated login attempts, a limitation discussed in Section 3.5. As displayed in Table 5, we found 273 domains exhibiting typo-tolerance during logins. This population of sites is significantly larger than previously documents, as discussed in Section 2. The most commonly accepted typos were related to letter casing (typos 1–3 in Table 5), accepted by between 251–264 domains. However, we saw 23 domains supporting other typos (none accepted an extra character at the front of the password).

A mean of 3 typo classes were accepted per domain, with at most 5. The most common overall typo-tolerance policy was accepting all three letter-casing typos (typos 1–3), employed by 241 domains (88% of typo-tolerant domains). The next most common policies were handling typos 1 and 2, and allowing typos 5 and 6, each appearing on 9 sites.

Differences across Rankings. We observed a slightly higher rate of typo-tolerance amongst top-ranked sites compared to lower-ranked ones. Within the top 10K, 4.9% of the analyzed domains accepted typos, compared to 2.3% within the top 100K and 1.3% over the top 1M.

Case Study: Case-Insensitive Passwords. We investigated the most common typo-tolerant policy that accepts any letter casing errors. We randomly sampled 30 such domains, manually verifying their typo-tolerance policies, and inspected the webpage code, as well as dynamically monitored website behavior while entering and submitting the password. We observed that when entering the real password with incorrect letter casing, the password value (as displayed and as stored in the HTML input element) was not modified. Upon form submission, we observe that the password was still transmitted without change. Thus, we did not find any client-side handling of the password letter casing. Instead, this policy is seemingly implemented server-side, likely by server functionality that modifies password letter casings (e.g., lowercasing/uppercasing all password letters). We found examples of such behavior in existing WordPress plugins [38].

Case Study: Pinterest. While prior work [11, 12] has referenced Facebook/Meta as the sole example of a top site employing typo-tolerant password authentication, we uncovered that Pinterest is another top site (ranked 33rd in the Feb. 6, 2023 snapshot of the Tranco top list [15]). Our analysis concluded that Pinterest accepts incorrect casing for the first password character (if a letter), inverted letter casing throughout the password (due to caps lock), and adding an extra character at the end of the password. We manually validated that Pinterest indeed supports this typo-tolerant policy (and again did not observe client-side implementation of typo-tolerance).

	Wrong Username	Wrong Password	Vuln to User Enum	Total Analyzed
10K	41	38	46	172
100K	308	362	418	2125
1M	3284	5343	5906	31,190

Table 6: The number of domains vulnerable to user enumeration, revealing either that a username is wrong, or that the password is wrong for a valid username.

4.7 Login Failures: User Enumeration

We collected and analyzed the login failure messages for 31.2K domains, failing to capture the messages on the remaining domains (in some cases, due to CAPTCHA errors encountered, as discussed in Section 3.5, and in other cases, we found that there was no message). We uncovered 5.9K domains (19%) exhibiting some form of user enumeration vulnerability, as shown in Table 6. Nearly 3.3K domains reveal that a username/email does not exist when given an incorrect one (rather than reveal that the username/password combination is incorrect, which would not leak if the username exists). Meanwhile, 5.3K domains reveal that the password is incorrect if the username/email exists for an account, allowing an attacker to learn that the username/email is correct. Thus, a sizable portion of domains are vulnerable to user enumeration.

Differences across rankings. We observe that higher-ranked sites were more likely to exhibit user enumeration vulnerabilities compared to lower-ranked ones. For top 10K domains, 26.7% had login failure messages allowing for user enumeration, compared to 19.7% for top 100K domains and 18.4% for top 1M sites. We hypothesize that as higher-ranked

sites often have larger user bases compared to lower-ranked domains, they are more motivated to provide user feedback upon login failures, often doing so in a vulnerable fashion.

Case Study: WordPress Login Failure Messages. We looked at the most common login error messages vulnerable to user enumeration, as shown in Appendix Table 8. We found that over 18% of domains shared the same message leaking username correctness when given the wrong password. We discovered that this message is the default for WordPress [39], even on the latest version (v.6.1.1). Likewise, we found evidence [40] that the top message leaking that a username is wrong, appearing on 4.4% of sites, is from the popular WordPress ProfilePress plugin [41] (with over 300K installations). Thus, popular web platforms like WordPress can heavily influence the user enumeration vulnerability of websites.

Case Study: Sensitive Websites. Prior work [30, 31] surveyed participants on sensitive online accounts, finding that most had accounts on sites that they would not want others to know about. Common site categories were adult, dating, and financial sites. We classified the website categories for the domains found vulnerable to user enumeration attacks, using SimilarWeb [42]. While SimilarWeb classified only 38% of domains, we still found 98 adult, 75 financial, and 24 dating domains vulnerable to user enumeration, highlighting a set of sites where user enumeration may be especially concerning.

4.8 Login Failures: Rate Limiting

On the 37.3K domains where we successfully created a test account for the rate limiting experiment, we were able to fully execute our experiment for 18.0K domains without error. For the other domains, our automated login process failed on one of the attempts due to CAPTCHA-related issues, preventing us from correctly assessing a site's rate limiting policy (a limitation discussed in Section 3.5). Note that the 18.0K domains successfully evaluated were not only those that were without CAPTCHAs; 4.5K (25%) displayed CAPTCHAs during the experiment that we were able to consistently solve correctly.

Across the 18.0K domains successfully evaluated, we observed 4335 domains (24%) demonstrating some rate limiting measure. For 2575 domains (14%), the site began blocking our login attempt by redirecting away from the login page or disabling the login page/form. For another 1760 (10%) domains, we were able to complete our sequence of failed login attempts, but the site blocked our subsequent valid login attempt, signaling that these domains rate limited the login attempts or locked the accounts. A quarter of these 1760 sites (456) continued to block our valid login an hour after the consecutive failed logins.

For the domains whose rate limiting policy was correctly assessed, we see that only a minority rate limit logins. However, given that we failed to analyze 52% of domains on which we created an account, the true prevalence may be higher. To better understand the domains we failed to evaluate, we randomly sampled and re-evaluated the rate limiting policy of 10 domains, this time solving CAPTCHAs manually. We found that only 3 domains rate limited our logins (with the other 7 domains allowing immediate login after our consecutive login attempts), a proportion commensurate with that observed on correctly tested domains. Thus, our results indicate that login rate limiting's deployment, as recommended by modern guidelines [1, 2], is still limited to a minority of sites.

Differences across Rankings. Among domains successfully analyzed, we observe that rate limiting logins was more prevalent amongst higher-ranked domains; 33.6% of the domains in the top 10K and 26.7% of domains in the top 100K demonstrated rate limiting, compared to the 24.1% for domains in the top 1M. Thus, top-ranked sites appear more likely to employ rate limiting, likely due to a combination of increased risk of authentication attacks, more valuable accounts, and more resources for implementing and supporting rate limiting (both computational and human resources).

5 Comparison to Prior Work

Here we summarize the prior work on similar aspects of website login policies, and compare their results with ours.

5.1 HTTPS Usage during Logins

Bonneau and Preibusch [43] analyzed TLS deployment on 45 websites in 2010, observing that 64% did not employ TLS during account signup, and less than half (44%) did so during account login. Parallel work [4] expanded examination to 150 domains, finding only 39% of the domains consistently submit passwords over TLS, and 41% did not use TLS at all.

Since these studies, HTTPS adoption has burgeoned [44]. Now 13 years later, one might hope for universal HTTPS adoption on any security-sensitive site, including those handling account signup and logins. We indeed found that TLS use across login pages is significantly higher than what prior work identified, although a non-trivial number of websites still exhibit insecure HTTPS configurations for login pages.

5.2 Password Entry

To our knowledge, prior work has not analyzed the prevention of copy-pasting on website logins in practice. Thus, our study provides the first data on this policy.

5.3 Storage of Plaintext Passwords

In 2010, Bonneau and Preibusch [4] also evaluated plaintext password storage on 150 domains, finding 16 domains emailing passwords upon registration and 36 doing so after password resets. In 2015, Bauman et al. [7] performed a similar manual analysis of 398 sites within the Alexa top 500, finding 11 cases of sites emailing plaintext passwords. In comparison, we analyzed the emails sent for accounts on 45K domains and conducted password resets on 27K sites. In total, we observed 570 domains emailing plaintext passwords.

5.4 Typo-Tolerant Password Authentication

Prior work on typo-tolerant password authentication [11, 12] highlights Facebook as the prime example of a typo-tolerant website [45, 46], while also referencing past reports of typo-tolerance on a couple other sites [47, 48]. However, we are not aware of prior work measuring the prevalence of typo-tolerance at scale. Thus, our study provides the first data on this practice, finding its use on hundreds of sites.

5.5 User Enumeration Vulnerabilities

Bonneau and Preibusch [4] examined login errors on 150 domains, finding 19% vulnerable to user enumeration. More recently in 2019, Hasegawa et al. [30,31] evaluated 87 popular and sensitive domains for user enumeration attacks, observing that two-thirds were vulnerable. In comparison, our study evaluates user enumeration across 31K sites, finding that a fifth of domains were vulnerable, similar to Bonneau and Preibusch [4], and identify sensitive domains affected.

5.6 Login Rate Limiting

In 2016, Golla et al. [5] manually analyzed rate limiting for 12 domains, performing 25 consecutive incorrect logins followed by a correct one. They observed rate limiting on 11 of 12 domains, with 10 domains locking the user account. Bonneau and Preibusch [4] also evaluated rate limiting on 150 domains using a script that attempted 100 consecutive incorrect logins. They found that the majority (127) did not employ rate limiting. More recently, Lu. et al. [6] performed similar rate limiting evaluations on 182 domains in the Alexa Top 500, finding 131 domains did not rate limit logins. In our study, we successfully evaluate the rate limiting policy of 18K sites, observing similarly that rate limiting is done by a minority ($\sim$25–30%) of sites.

6 Concluding Discussion

In this study, we conducted the largest evaluation of website login policies to date, assessing 18K–359K sites across various policy components (orders of magnitude more than prior work). Our results establish empirical grounding on the state of modern web authentication, characterizing the insecure login policies that occur. Here, we synthesize our findings into lessons for improving web authentication moving forward.

Importance of Large-Scale Web Authentication Measurements. By measuring at scale, we uncovered unique aspects of online authentication, including varying login policies deployed by sites across different ranking ranges. For example, while there was scant evidence of typo-tolerance use in practice (Section 5.4), we found hundreds of sites deploying typo-tolerance, and identified that policies primarily account for letter casing typos (Section 4.6). This finding has security implications as recent work [12] identified that typo-tolerant schemes, while more usable, also are significantly more vulnerable to credential stuffing attacks. These newly identified domains, including a top 50 site, motivate further investigation into typo-tolerance's security-usability tradeoff. We also observed certain subpopulations exhibiting insecure practices, such as poor HTTPS configurations by government and educational sites (Section 4.2), and frequent disabling of login form pasting by top Indian domains (Section 4.3). Having identified these subpopulations, community efforts to inform affected parties and encourage remediation can drive real improvements in online authentication.

However, conducting web authentication measurements at scale is extremely challenging given the web's heterogeneity and the extensive engineering required for automation. Our data and results will eventually become stale, and later investigations into web authentication will be needed. Future work should refine existing measurement methods, such as by developing more accurate classifiers or identifying ways to better manage web diversity.

Influence of Popular Web Frameworks. We discovered that several login security issues arose due to implementation decisions by popular web frameworks. For example, about a fifth of domains vulnerable to user enumeration appear to simply use WordPress's default login failure messages (Section 4.7). Similarly, the most common typo-tolerance policy likely arose due to how popular server-side software modified passwords (Section 4.6). While these web frameworks may be the cause of prevalent authentication issues, they can also be the source of solutions. Software updates that address authentication concerns could drastically reduce vulnerable populations. Meanwhile, if popular web frameworks supported recommended practices by default, such as rate limiting (deployed by only a minority of sites, as seen in Section 4.8), we would likely observe significantly higher adoption levels.

Improving the Web HTTPS Ecosystem. As discussed in Section 5.1, our analysis of HTTPS use during web logins indicates that TLS use has significantly improved compared to prior observations. Community efforts to drive wider TLS adoption have led to more secure online logins at large. However, work remains as some sites still use insecure communication channels. From Section 4.2, we found that mixed content remains a relevant issue on login pages, and given the prominence of websites still supporting login pages over HTTP, further effort is needed to incentivize website operators to migrate all sensitive pages (e.g., login pages) to HTTPS only. In many cases, we hypothesize that the operators are not aware that login pages are still available over HTTP, or are unaware of the consequences of hosting HTTP login pages. Here, outreach campaigns may be effective at reducing this population (as done by prior work in other security contexts [49–51]).

Combining Crowdsourced and Automated Approaches. Our case study of the crowdsourced Plain Text Offenders (PTO) project (in Section 4.5) highlights how even popular crowdsourcing efforts have limited coverage (as we discovered hundreds of new sites storing plaintext passwords) and must deal with stale data (as many domains in the PTO list no longer belong). Especially as such projects partly serve to

"name and shame" misbehaving sites, outdated data can unnecessarily besmirch sites that take corrective actions. However, such efforts can identify insecure cases that automated methods cannot, and thus are highly complementary approaches. We advocate for hybrid efforts moving forward, where crowdsourced data can be augmented by automated measurements, as well as fed into the automation to allow for periodic re-evaluations (producing fresh results).

Driving Adherence to Modern Standards. While modern standards [1–3, 9] provide guidelines for secure and usable policies at each login stage (except with typo-tolerance, which warrants further investigation and incorporation into guidelines), we observe nonadherence by sizable populations of sites throughout our findings. Further investigation is needed into why website operators do not adopt these standards, to determine the best avenues for affecting change. One possible lever for driving improved authentication practices could be through enforcing regulations. For example, we observed in Section 4.5 that many domains we that expect are storing plaintext passwords are in the European Union, where GDPR could be leveraged to penalize such insecure practices. Such enforcement could raise awareness and incentivize remediation of insecure website behaviors.

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] Paul Grassi, Michael E Garcia, and James L Fenton. Digital identity guidelines. *NIST Special Publication*, 800:63–3, 2017.

[2] National Cyber Security Centre (NCSC). Password administration for system owners. *NIST Special Publication*, 2018.

[3] Open Web Application Security Project. Authentication Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

[4] Joseph Bonneau and Sören Preibusch. The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In *Workshop on the Economics of Information Security (WEIS)*, 2010.

[5] Maximilian Golla, Theodor Schnitzler, and Markus Dürmuth. "Will Any Password Do?" Exploring Rate-Limiting on the Web. *Who Are You?! Adventures in Authentication Workshop (WAY)*, 2016.

[6] Bo Lu, Xiaokuan Zhang, Ziman Ling, Yinqian Zhang, and Zhiqiang Lin. A measurement study of authentication rate-limiting mechanisms of modern websites. In *Annual Computer Security Applications Conference (ACSAC)*, 2018.

[7] Erick Bauman, Yafeng Lu, and Zhiqiang Lin. Half a century of practice: Who is still storing plaintext passwords? In *International conference on information security practice and experience*, 2015.

[8] Chrome Developers. Chrome UX Report. URL: https://developer.chrome.com/docs/crux/.

[9] National Cyber Security Centre. Let them paste passwords, 2017. URL: https://www.ncsc.gov.uk/blog-post/let-them-paste-passwords.

[10] Google Cloud. Modern password security for system designers. URL: https://cloud.google.com/solutions/modern-password-security-for-system-designers.

[11] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. pASSWORD tYPOS and how to correct them securely. In *IEEE Symposium on Security and Privacy (SP)*, 2016.

[12] Sena Sahin and Frank Li. Don't Forget the Stuffing! Revisiting the Security Impact of Typo-Tolerant Password Authentication. In *ACM Conference on Computer and Communications Security (CCS)*, 2021.

[13] Joe DeBlasio, Stefan Savage, Geoffrey M Voelker, and Alex C Snoeren. Tripwire: Inferring internet site compromise. In *ACM Internet Measurement Conference (IMC)*, 2017.

[14] Kostas Drakonakis, Sotiris Ioannidis, and Jason Polakis. The cookie hunter: Automated black-box auditing for web authentication and authorization flaws. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[15] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Network and Distributed Systems Security Symposium (NDSS)*, 2019.

[16] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[17] Scikit-learn. sklearn.svm.SVC. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[18] ScraperAPI. The Proxy API For Web Scraping. URL: https://www.scraperapi.com/.

[19] Ding Wang and Ping Wang. The emperor's new password creation policies. In *European Symposium on Research in Computer Security (ESORICS)*, 2015.

[20] Kevin Lee and Sten Sjöberg and Arvind Narayanan. Password policies of most top websites fail to follow best practices. In *Symposium on Usable Privacy and Security (SOUPS)*, 2022.

[21] Tobias Seitz, Manuel Hartmann, Jakob Pfab, and Samuel Souque. Do differences in password policies prevent password reuse? In *SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2017.

[22] Daniele Faraglia. Welcome to Faker's documentation! URL: https://faker.readthedocs.io/.

[23] AZcaptcha. Auto Captcha Solver Service and Cheap Captcha Bypass Service Provider - AZcaptchas. URL: https://azcaptcha.com/.

[24] Selenium. Actions API - Selenium. URL: https://www.selenium.dev/documentation/webdriver/actions_api/.

[25] Michal Danilk. langdetect 1.0.9. URL: https://pypi.org/project/langdetect/.

[26] Terry Yin. translate 3.6.1. URL: https://pypi.org/project/translate/.

[27] Hugging Face. sentence-transformers/all-distilroberta-v1 - Hugging Face. URL: https://huggingface.co/sentence-transformers/all-distilroberta-v1.

[28] Selenium. URL: https://www.selenium.dev/.

[29] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. Re: CAPTCHAs—Understanding CAPTCHA-Solving Services in an Economic Context. In *USENIX Security Symposium*, 2010.

[30] Ayako Akiyama Hasegawa, Takuya Watanabe, Eitaro Shioji, and Mitsuaki Akiyama. I know what you did last login: inconsistent messages tell existence of a target's account to insiders. In *Annual Computer Security Applications Conference (ACSAC)*, 2019.

[31] Ayako Akiyama Hasegawa, Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Tatsuya Mori. Addressing the Privacy Threat to Identify Existence of a Target's Account on Sensitive Services. *Journal of Information Processing*, 28:1030–1046, 2020.

[32] Asuman Senol, Gunes Acar, Mathias Humbert, and Frederik Zuiderveen Borgesius. Leaky Forms: A Study of Email and Password Exfiltration Before Form Submission. In *USENIX Security Symposium*, 2022.

[33] US Department of Education. Family Educational Rights and Privacy Act (FERPA). URL: https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html.

[34] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. Neither Snow Nor Rain Nor MITM...: An Empirical Analysis of Email Delivery Security. In *ACM Internet Measurement Conference (IMC)*, 2015.

[35] Intersoft Consulting. Art. 32 GDPR Security of processing. URL: https://gdpr-info.eu/art-32-gdpr/.

[36] Ionut Ilascu. First GDPR Sanction in Germany Fines Flirty Chat Platform EUR 20,000. URL: https://www.bleepingcomputer.com/news/security/first-gdpr-sanction-in-germany-fines-flirty-chat-platform-eur-20-000/.

[37] Aviem Zur Igal Tabachnik, Omer van Kloeten. Plain text offenders. URL: https://plaintextoffenders.com/.

[38] Lew Ayotte. Case-Insensitive Passwords. URL: https://wordpress.org/plugins/case-insensitive-passwords/.

[39] WordPress Foundation. Build a Site, Sell Your Stuff, Start a Blog & More. URL: https://wordpress.com/.

[40] Osama Ibrahim. Unable to login with email address as the username - WordPress. URL: https://wordpress.stackexchange.com/questions/374158/unable-to-login-with-email-address-as-the-username-wordpress.

[41] ProfilePress. Modern Ecommerce, User Profile & WordPress Membership Plugin. URL: https://profilepress.com/.

[42] SimilarWeb. URL: https://similarweb.com/.

[43] Joseph Bonneau and Sören Preibusch. The privacy jungle: On the market for data protection in social networks. In *Economics of information security and privacy*, pages 121–167. Springer, 2010.

[44] Google. HTTPS encryption on the web. URL: https://transparencyreport.google.com/https/overview?hl=en.

[45] Emil Protalinski. Facebook passwords are not case sensitive, 2011. URL: www.zdnet.com/article/facebook-passwords-are-not-case-sensitive-update/.

[46] Josh Hendrickson. Is Vanguard Making It Too Easy for Cybercriminals to Access Your Account?, 2019. URL: https://www.howtogeek.com/402761/facebook-fudges-your-password-for-your-convenience/.

[47] Dylan Tweney. Amazon.com Security Flaw Accepts Passwords That Are Close, But Not Exact, 2011. URL: https://www.wired.com/2011/01/amazon-password-problem/.

[48] Zack Whittaker. Surprise! Your online banking password might not be as secure as you thought, 2017. URL: https://www.zdnet.com/article/surprise-online-bank-passwords-may-not-be-case-sensitive/.

[49] Eric Zeng, Frank Li, Emily Stark, Adrienne Porter Felt, and Parisa Tabriz. Fixing HTTPS misconfigurations at scale: An experiment with security notifications. In *Workshop on the Economics of Information Security (WEIS)*, 2019.

[50] Max Maass, Alina Stöver, Henning Pridöhl, Sebastian Bretthauer, Dominik Herrmann, Matthias Hollick, and Indra Spiecker. Effective Notification Campaigns on the Web: A Matter of Trust, Framing, and Support. In *USENIX Security Symposium*, 2021.

[51] Frank Li, Grant Ho, Eric Kuan, Yuan Niu, Lucas Ballard, Kurt Thomas, Elie Bursztein, and Vern Paxson. Remedying web hijacking: Notification effectiveness and webmaster comprehension. In *International Conference on World Wide Web (WWW)*, 2016.

[52] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text mining: Applications and Theory*, 1:1–20, 2010.

[53] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

A Keywords Extraction Details

To identify relevant keywords and phrases for our method, we manually analyzed a random sample of domains and applied the ranking algorithm Term Frequency-Inverse Document Frequency (TF-IDF) [16]. TF-IDF ranks the importance of a word by its frequency in a document, logarithmically scaled by the number of documents it occurs in. The top words and phrases produced from the method are then manually investigated and keywords are selected. We initially explored two other approaches, Rapid Automatic Keyword Extraction (RAKE) [52] and Text Rank [53], but found TF-IDF to be the most effective.

B Manual Evaluation of Method Performance

To better understand how our method (from Section 3) performs, we investigate a random sample of domains and manually evaluate how our method executed on these domains at each stage of the measurement. We conduct our evaluation in two phases. In the first phase, we randomly sample 100 domains from our full population of 1M domains, and evaluate our method's steps to find forms and create accounts. As our method ultimately created accounts on a small fraction of sites, we expect that only a small number of domains in the first phase's sample will have successfully created accounts, inhibiting meaningful evaluation of our method's login policy evaluation. Thus, in a second phase, we randomly sampled 75 domains where our method successfully created an account, and evaluate how well it inferred the login policies.

B.1 Phase 1: Finding Forms and Creating Accounts

In this first phase, we focus on our method's ability to find the account login, signup, and password reset forms, and successfully complete the account creation workflow.

- **Account Signup Forms + URLs:** We manually found 40 out of 100 domains providing a URL for account creation. Our method missed 23 of these domains (FNR = 58%). Of these, 9 domains had multi-stage account creation workflows (which our method was not designed to handle), and 8 domains lacked the relevant keywords in their signup URL or form (Section 3.2.2). The remaining cases involved the signup page not being in our method's site search exploration (i.e., deeper within a site than we explored), false negatives by our form classifier, or user interaction required

before displaying the form. Our method also had 6 false positives (FPR = 10%) due to the presence of relevant keywords in URLs or page forms (but this does not result in false positives during account creation, as discussed below).

- **Login Forms + URLs:** We manually found 54 domains with login URLs (note, this number is higher than the number of domains with account signup URLs as a number of sites do not seem to provide public account creation, such as educational and financial institutions). Our method missed the login page on 17 domains (FNR = 31%). The most common issue (8 domains) was a lack of relevant keywords in the URL or login form. The remaining domains either embedded the login page further within a site than our method searched, required user interaction before displaying the login form, deployed multi-step login forms, or caused a false negative classification from our form classifier. Here we had no false positives.
- **Password Reset Forms + URLs:** We manually found 25 domains with password reset URLs. Our method missed 22 domains (FNR = 88%), of which the majority (13) were due to missing keywords in the URL or form. We also found 5 domains with reset forms requiring user interaction before displaying. The remaining issues were reset URLs not in our method's site search exploration, multi-stage forms, and non-form-based resets. Here, we had 1 false positive (FPR = 1%) due to the presence of relevant keywords (but a false positive would not result in a false positive password storage policy inference later on).
- **Account Creation:** Of the 17 domains where we found the signup URL, we successfully created accounts on 3 domains (with 14 false negatives). We note that several sites required phone or ID verification, which is not handled by our method. For 6 false negative domains, we provided non-accepted values during the form filling process. Other errors included CAPTCHA-solving failures and misclassifications by the signup success classifier. We observed no false positives (where we believed we had created an account but had not).

From our analysis above, we believe that a promising direction for improving our automated method in the future would be to implement support for identifying and completing multi-stage forms. In addition, keyword heuristics for forms and URLs can be further refined, and other page features could be incorporated for form detection (which would likely aid with multi-stage form detection). We also struggled with providing acceptable form values, so future work can improve upon the classification and completion of form fields.

B.2 Phase 2: Login Policy Evaluation

To evaluate how well our method inferred login policies, in this second phase, we manually analyze a sample of 75 domains which our method successfully created accounts on.

- **Login Page Visit / Password Transmission:** These policies only required finding the login page, which our method did so with high accuracy (see Appendix B.1). We did not identify false positive or false negative policy inferences on these sites.
- **Account Credential Entry:** Upon manual investigation, we did not find any of the 75 domains disabling copy-pasting. Our method likewise did not exhibit any false positive or false negative detections. Note that we randomly sampled 20 domains that our method detected as disabling copy-pasting, and manually confirmed that these inferences were correct (no false positives). Thus, our policy inference here is highly accurate.
- **Password Storage/Retrieval:** We manually found that 2 of the 75 domains sent a plaintext password in an email upon account registration. Our method observed the same on both domains as well. However, we manually found that 1 domain sent a plaintext password during a password reset, which our method missed as it did not find that domain's password reset URL (see Appendix B.1). Thus, as discussed in Section 4.5, the number of sites that our method detected storing plaintext passwords is a lower bound.
- **Password Validation:** Our method fully evaluated typo-tolerance on 29 (39%) of the domains. On the other domains, our automated CAPTCHA solving failed for one of the repeated attempts. Upon manual investigation, we found 1 out of 75 domains exhibiting a typo-tolerant policy, which was also correctly detected by our method. There were no false positives where our method incorrectly inferred typo-tolerance for a site. Thus, our results provide a lower bound on sites deploying typo-tolerance.
- **User Enumeration:** Upon manual investigation, we found 23 domains exhibiting a user enumeration vulnerability in their login error message. Our method correctly extracted error messages on 55 out of 75 domains (73%), and correctly identified 16 domains exhibiting user enumeration issues. The 7 missing domains (with user enumeration issues) did not have their error message extracted correctly due to a lack of relevant keywords or the error message appearing outside of the login form. Our method also misclassified 5 domains as false positives, thus as discussed in Section 3.3.6, our results provide an approximation of the prevalence of this vulnerability.
- **Rate Limiting:** We evaluated rate limiting through the same experiment as described in Section 3.3.7, but performing the repeated logins manually within a browser. Note that as we conducted our logins manually, rate limiting results can differ, such as due to anti-bot/anti-automation mechanisms or detection of organic human behavior. In total, we detected rate limiting on 15 domains (20%).

  Our method fully analyzed the rate limiting policy of 30 domains (40%), failing on other domains due to our automated CAPTCHA solver failing during one of the login attempts. Among these missed domains were 11 manually found to do rate limiting. Of the successfully-analyzed domains, our method found 6 domains deploying rate limiting.

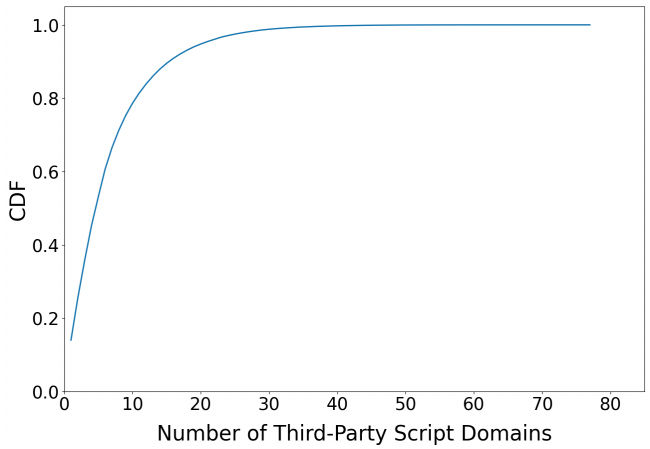

Figure 3: CDF of the number of unique third-party domains that script resources are loaded from on the login pages of our 358.9K investigated sites.

Four of these domains were confirmed manually. For the remaining two, we believe that rate limiting did occur, but likely due to anti-bot/anti-automation mechanisms that did not trigger during our manual login attempts. While we were unable to analyze rate limiting across all domains, our findings in Section 4.8 are commensurate with those from this manually-analyzed sample. Thus we believe our rate limiting observations hold generally (as discussed also in Section 4.8).

Improving the discovery of login pages and password reset forms, as well as automated account creation, would naturally expand our password policy evaluation (as discussed in Appendix B.1). Beyond those directions, improved automated CAPTCHA solving would significantly enhance our policy analysis, as CAPTCHA solving issues hindered our analysis of password validation and rate limiting on many sites. Related, more steathy web crawling may reduce the number of CAPTCHAs encountered, similarly improving policy inference. Finally, future work can develop a more effective user enumeration vulnerability classifier, as our current model exhibits a high false positive rate.

C Third-Party Scripts on Login Pages

Prior work uncovered third-party JavaScript on login pages exfiltrating credentials from login forms, primarily for tracking, marketing and analytic purposes [32]. Here, we perform an initial analysis of the extent to which login pages are potentially vulnerable due to third-party script inclusion. For the login pages we identified, we extract the *src* attribute of all script elements. We then characterize whether the script is loaded from the same domain as the login page (i.e., first-party script) or another domain (i.e., third-party script).

Out of the 358.9K sites with login pages collected, we found that the vast majority (98.3%) of login pages included at least one script tag. In total, 296.9K (82.7%) sites had a login page with a third-party script. Among these sites,

Rank	Script Domain	# Sites (% Sites)
1	googletagmanager.com	181,597 (50.6%)
2	google-analytics.com	180,306 (50.2%)
3	connect.facebook.net	105,514 (29.4%)
4	googleads.g.doubleclick.net	60,707 (16.9%)
5	gstatic.com	55,318 (14.4%)
6	google.com	49,650 (13.8%)
7	ajax.googleapis.com	44,827 (12.5%)
8	cdnjs.cloudflare.com	37,387 (10.4%)
9	cdn.jsdelivr.net	27,478 (7.7%)
10	static.hotjar.com	23,265 (6.5%)

Table 7: The 10 most common third-party script domains observed on the login pages of our 358.9K analyzed sites.

Figure 3 depicts the CDF of the number of distinct third-party script domains on login pages. We found that login pages loaded scripts from a median of 4 third-party script domains, and about a fifth of login pages had at least 10 third-party script domains. The sites with the most diverse third-party script sources (up to 77 domains) were primarily e-commerce and online services.

In total, there were 155K unique third-party script domains. The 10 most common script domains are shown in Table 7. We found that the most common script domains were related to analytics and advertisement services (e.g., from Google), e-commerce (e.g., Shopify), content distribution networks (CDNs), social media (e.g., Facebook, Tiktok), and search engines. The remaining script domains exhibit a long-tail distribution, with 145K script domains (93.4%) observed on five or fewer sites, and 129K (82.9%) domains observed on only one site.

Overall, the vast majority of login pages are potentially vulnerable to credential exfiltration by malicious third-party scripts (although we leave an exploration of how third-party scripts truly interact with login credentials for future work).

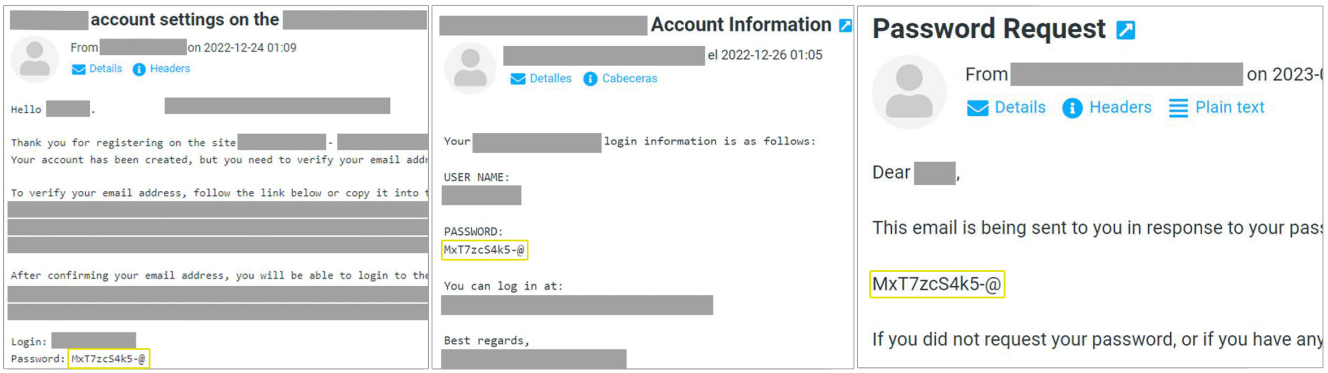

Figure 4: Examples of emails received with plaintext passwords upon account registration, after email verification, and after password reset.

Rank	Wrong Username	% Sites	Wrong Password	% Sites
1	Unknown email address. Check again or try your username.	4.4%	Error: The password you entered for the email address [Email] is incorrect. Lost your password?	18.1%
2	Incorrect email address specified	0.7%	Incorrect Password	1.2%
3	User does not exist.	0.6%	Wrong password	0.9%
4	You have specified an incorrect username. Please check your username and try again. If you continue to have problems please contact the Board Administrator.	0.5%	Invalid password	0.7%
5	Username not found.	0.5%	Incorrect password. Please try again.	0.7%

Table 8: The top 5 login failure messages observed across analyzed sites that allow for user enumeration, either by revealing a username is wrong, or that a password is wrong (but the username is correct).