Don't Forget the Stuffing! Revisiting the Security Impact of Typo-Tolerant Password Authentication

Sena Sahin
ssahin8@gatech.edu
Georgia Institute of Technology

Frank Li
frankli@gatech.edu
Georgia Institute of Technology

ABSTRACT

To enhance the usability of password authentication, typo-tolerant password authentication schemes permit certain deviations in the user-supplied password, to account for common typographical errors yet still allow the user to successfully log in. In prior work, analysis by Chatterjee et al. demonstrated that typo-tolerance indeed notably improves password usability, yet (surprisingly) does not appear to significantly degrade authentication security. In practice, major web services such as Facebook have employed typo-tolerant password authentication systems.

In this paper, we revisit the security impact of typo-tolerant password authentication. We observe that the existing security analysis of such systems considers only password spraying attacks. However, this threat model is incomplete, as password authentication systems must also contend with credential stuffing and tweaking attacks. Factoring in these missing attack vectors, we empirically re-evaluate the security impact of password typo-tolerance using password leak datasets, discovering a significantly larger degradation in security. To mitigate this issue, we explore machine learning classifiers that predict when a password's security is likely affected by typo-tolerance. Our resulting models offer various suitable operating points on the functionality-security tradeoff spectrum, ultimately allowing for partial deployment of typo-tolerant password authentication, preserving its functionality for many users while reducing the security risks.

CCS CONCEPTS

• **Security and privacy → Authentication**.

KEYWORDS

Password Authentication; Security Analysis; Machine Learning

ACM Reference Format:
Sena Sahin and Frank Li. 2021. Don't Forget the Stuffing! Revisiting the Security Impact of Typo-Tolerant Password Authentication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3460120.3484791

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea.
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484791

1 INTRODUCTION

Despite well-documented security and usability concerns [5], human-chosen passwords remain the *de facto* standard mechanism for authentication across the web. As a consequence, the security of much of the web ecosystem and its billions of users remains dependent on how users and websites manage password authentication.

One password usability issue that arises in practice is that users struggle to correctly type in their passwords, making typographical errors that result in delayed or failed login attempts [18, 19]. To address this issue, *typo-tolerant* password authentication systems accept submitted passwords that deviate in certain regards from the true password, essentially correcting for common classes of typographical errors. Several major web services, notably including Facebook, have been documented as having deployed typo-tolerance for online logins [2, 15, 34].

While typo-tolerance during password authentication may improve password usability, one might suspect that it also degrades security during online password guessing attacks, as each password guessed by the attacker covers a range of passwords rather than a single one. In 2016, Chatterjee et al. [7] provided the first systematic usability and security analysis of typo-tolerant password authentication. They identified that typographical mistakes made by users during password entry were frequent, with several common errors that could be readily corrected with simple password transformation functions. As a result, typo-tolerance would indeed significantly improve password usability. Furthermore, they also formally and empirically analyzed the impact of typo-tolerance on password security under a specific threat model, finding that the impact was minimal. Thus, the work concluded that typo-tolerance does not actually result in a tradeoff between security and usability, but rather produces notable usability gains with negligible degradation in security.

In this paper, we re-examine the security impact of typo-tolerant password authentication. Our exploration is driven by the observation that the existing analysis by Chatterjee et al. [7] used an online attack model that effectively considered only password spraying attacks [22], where attackers guess common passwords (potentially based on some password probability distribution). We identify that this threat model is incomplete. In addition to password spraying attacks, web services also contend with credential stuffing attacks [21, 24], where attackers leverage leaked passwords from one website to target accounts on other services. Numerous websites have suffered high-profile credential stuffing attacks [1, 24, 30], demonstrating that this threat is salient in practice. Additionally, prior research [10, 25, 36] has developed credential tweaking attacks, where beyond only "stuffing" leaked passwords, attackers also generate variants of leaked passwords for login attempts.

Credential stuffing exploits the high password reuse rate among users. However, many users who do not reuse the same password across websites often use minor variations for different sites [10]. We hypothesize that for many such users, the differences between two sites' passwords could be corrected by the policies applied in typo-tolerant password authentication schemes, thus allowing for successful credential stuffing (and tweaking) attacks even when the leaked passwords used by the attacker are not exact matches.

We empirically evaluate our hypothesis by analyzing two recent password leak datasets containing nearly 4 billion account credentials. We first assess the impact of typo-tolerance on password spraying attacks, reproducing the prior finding that typo-tolerance results in limited additional attack success. Even under a liberal typo-tolerance policy where attackers are afforded 1000 online guesses, less than 1% of emails become newly vulnerable to password spraying after enabling typo-tolerance. We then evaluate credential stuffing and tweaking attacks, finding that typo-tolerance can exacerbate the effectiveness of these attacks significantly. In the worst case, as many as a third of emails become newly vulnerable to credential stuffing and tweaking after enabling typo-tolerance, and up to 9% of emails under more realistic settings.

Our findings reveal that the security consequences of typo-tolerant password authentication are more severe than previously understood. However, this conclusion does not necessarily mean that typo-tolerance must be wholesale abandoned. To determine whether typo-tolerance can still be deployed to some extent, we explore machine learning classifiers that predict if a password may be newly susceptible to credential stuffing under typo-tolerance. Using one of our leak datasets, we develop proof-of-concept models that provide different recall versus false positive rate operating points that may be suitable for practical use. Ultimately, using such models in conjunction with typo-tolerant password authentication allows web services to manage the tradeoff between the functionality of typo-tolerance and its security concerns.

In summary, our paper makes the following contributions:

- We replicate the prior security analysis of typo-tolerant password authentication on password leak datasets that are two orders of magnitude larger, confirming the previous work's results continue to hold [7].
- We expand our current understanding of typo-tolerant password authentication by analyzing how it is impacted by credential stuffing and tweaking attacks, which are missing from the existing analysis's threat model [7]. Our findings indicate that the authentication security degradation from enabling typo-tolerance is significantly greater under this more complete threat model.
- We explore machine learning classifiers that can predict when security may degrade for a password under typo-tolerance. These classifiers offer different operating points trading off recall and false positives rates, allowing web services to still deploy typo-tolerant password authentication to a broad population while limiting the security risks.
- We discuss future directions for exploring how to further harden typo-tolerant password authentication.
- We have shared our research with Facebook, which has taken steps internally to address the concerns.

2 BACKGROUND AND RELATED WORK

In this section, we provide an overview of typo-tolerant password authentication, highlighting the most relevant details for our subsequent investigation. We also discuss related work on password attacks and defenses.

2.1 Typo-Tolerant Password Authentication

Typo-tolerant password authentication systems permit certain typographical errors in the submitted password while still authenticating the user. Such systems have been used in practice, such as by Facebook [15], to improve the user login experience. In 2016, Chatterjee et al. [7] conducted the first systematic usability and security analysis of such authentication schemes.

Usability Analysis. Through a crowdsourced user study and an analysis of telemetry from live login attempts at Dropbox, Chatterjee et al. identified several classes of common typos made during password entry. The five most common mistakes were, in decreasing order: 1) the case of all letters was flipped (likely due to the caps lock key), 2) the case of the first letter was flipped, 3) an extra character appeared at the password end, 4) an extra character appeared at the password start, and 5) the last character was not shift-key modified. From the Dropbox login telemetry, the researchers observed that 9.3% of the failed login attempts were due to these typos that could have been automatically corrected using string transformation functions (one for each common typo class). Had the authentication system tested such corrections on the submitted passwords before validation, nearly 20% of Dropbox users who made typos would have logged in faster. In addition, 3% of all users failed to log in altogether but would have succeeded had typo-tolerant checking been enabled for the three most common categories of typos. These results empirically demonstrated that typo-tolerance can notably improve authentication usability.

The specific password corrector functions identified were (in the same order as the common typo classes that they correct for):

(1) *swc_all*: Swap the case of all letters in the password.
(2) *swc_first*: Swap the case of the first password character.
(3) *rm_last*: Remove the last password character.
(4) *rm_first*: Remove the first password character.
(5) *n2s_last*: Shift-key modify the last password character.

The authors also defined typo-tolerance policies combining these corrector functions, with the C_{TopX} policy containing the first X correctors, in the order listed above (e.g., C_{Top3} contains *swc_all*, *swc_first*, and *rm_last*). In our work, we will consider the same set of corrector functions and typo-tolerance policies.

Security Analysis. While typo-tolerance allows users to more easily authenticate, one might expect that it could also allow attackers to more readily authenticate as a target, as incorrect attacker password guesses might be automatically corrected to the target's true password. Considering online password spraying attacks [22], where the optimal attacker strategy is to guess the most popular passwords (in decreasing order), Chatterjee et al. provided both a formal and empirical security evaluation of typo-tolerance, demonstrating that in fact, such attackers do not gain a significant advantage once typo-tolerance is enabled. Through simulating password spraying attacks using password dumps from three websites, the authors found that if the attacker was permitted 10 login

attempts per account (a reasonable quantity given rate-limiting and additional login defenses commonly deployed by online services), the attack success rate increases by only 0.27% when using the most permissive typo-tolerance policy. Thus, they conclude that typo-tolerance provides substantial authentication usability benefits without noticeable degradation in authentication security.

In this work, we revisit this prior security analysis of typo-tolerant password authentication by evaluating the security implications of such schemes under a more comprehensive threat model that, in addition to password spraying [22], includes practical attacks such as credential stuffing [21, 24] and credential tweaking [25]. We note that more recently, a personalized variant of typo-tolerance was developed [8] that customizes the typo corrections to the user. Blanchard et al. [4] also developed a typo-tolerance scheme with reduced server computation costs. In this study, we focus on traditional typo-tolerance in password authentication systems, as it has been used in practice [2, 15, 34], likely due to its simpler design (thus facilitating deployment).

2.2 Password Attacks and Defenses

In this study, we investigate the impact of password spraying [22], credential stuffing [21, 24], and credential tweaking [25] attacks on typo-tolerant password authentication, as well as hardening such typo-tolerant schemes. Prior work has also considered such password attacks and defenses in a more general context.

Password spraying is enabled by the passwords commonly used by many users. Meanwhile, credential stuffing and tweaking attacks are fueled by the widespread use of identical or similar passwords across online services, as extensively documented by prior research [10, 26, 27, 31, 32, 35]. As a consequence, data breaches at one service result in account compromise at other services.

Various defenses against these attacks have been proposed. For password spraying, one approach [16, 29] is to derive signals from the passwords used in failed login attempts to detect and block ongoing attacks. Password strength meters and blocklists also aim to discourage the use of common passwords [14, 23, 28]. To defend against credential stuffing attacks, Wang et al. [37] developed a cooperative framework where different web services can exchange telemetry to detect and block credential stuffing. Another line of defenses identifies if user credentials may be in leaked datasets, typically resulting in user notifications and password resets [9, 13, 20, 33]. The most sophisticated attack we consider is credential tweaking, where the attacker generates variants of a leaked password for their login attempts. In 2019, Pal et al. [25] used a generative neural network model to perform the state-of-the-art credential tweaking attack, improving upon the effectiveness of previously developed attacks [10, 36, 38]. To date, defenses against this attack class are limited to password meters that warn users against using susceptible passwords [25].

3 METHODOLOGY

In this section, we describe the datasets that we use to empirically evaluate the impact of typo-tolerance on password authentication security. Instead of assessing a live authentication system, which would entail user privacy, security, and ethical concerns, we rely on public password leaks, as similarly done in prior password research [7, 10, 25, 32].

3.1 Data Source

Our study relies on two recent and massive password leak datasets. These datasets are publicly available online, although we avoid publishing their specific locations here due to their sensitive nature. While verifying the data in any data leak is challenging due to ethical concerns, these datasets have been previously used by security researchers without raising validity concerns [1, 6, 17, 25].

BreachCompilation. This leaked dataset is an aggregation of various password breaches, found by 4iQ in late 2017 [6]. It contains 1.4B account credentials associated with 1.1B distinct emails and 463M unique passwords. While this dataset contains well-known breaches, including LinkedIn, Yahoo, and Myspace, it does not indicate which breach sourced a given credential.

Collection#1. This dataset [17], uncovered in early 2019, contains multiple compilations of password leaks. The dataset consists of 31 folders (listed in Table 6 in the Appendix), where the folder names hint at the breach origins (e.g., leaks from certain countries or certain types of online services). While certain folders contain files indicating the specific source of a leak (e.g., a domain name in the filename), many of the folders are aggregations themselves, again preventing us from ascertaining the provenance of the credentials. In total, the collection consists of 2.7B account credentials associated with 772M unique emails and 21M distinct passwords.

3.2 Data Processing

BreachCompilation. We parse the BreachCompilation dataset's 1.4B lines, which are formatted as delimiter-separated (account, password) pairs. As we will use email addresses to reliably associate account credentials with the same user, we filter out lines without a valid email address as the account identifier. We also filter out lines without a likely valid plaintext password, including those where the password is longer than 31 characters (many of which are hash values, and account for less than 0.1% of all passwords) or shorter than 3 characters. Finally, we eliminate the approximately 36.5K emails associated with over 100 passwords, which are unlikely to represent real users. In total, we removed 2.8M lines, accounting for 0.2% of the original data.

As listed in Table 6 in the Appendix, the final dataset consists of 1.1B unique emails, of which 182.1M emails are associated with multiple passwords. Of these multi-password emails, 21% are associated with the same password multiple times. This fraction of emails is commensurate with the password reuse rates observed in prior studies [10, 32]; thus we believe this observation reflects password reuse rather than the leak containing extensive duplicated data.

Collection#1. The Collection#1 dataset contains various file types and formats, including delimiter-separated text files, files with SQL statements, and HTML documents. We also observe files containing only hash values instead of plaintext passwords. We focus on processing the delimiter-separated text files, which account for 99.9% of the data.

After parsing these files (considering multiple potential delimiters), we process the resulting (account, password) pairs in a similar

fashion as with the BreachCompilation dataset. We filter out pairs without a valid email for the account, pairs where the password is longer than 31 characters or shorter than 3 characters, and emails with over a hundred associated passwords. In total, we filtered out 0.5% of the original data.

As listed in Table 6 in the Appendix, the Collection#1 dataset is organized into 31 separate folders, where each folder name suggests a common characteristic of the contained breach data. We observe similarities in folder names though (e.g., *EUcombos* and *EUcombos_1*), suggesting that the same breached data may be duplicated across multiple folders. Furthermore, we find extensive overlap in the account credentials between folders. Thus, we evaluate each folder as a separate leak compilation dataset (we will refer to each folder's data as a separate Collection#1 leak combination).

We further investigate the potential existence of duplicate data for each leak combination. Again, considering emails associated with multiple passwords, we calculate the fraction that are associated with the same password multiple times, seeing whether this rate is commensurate with the password reuse rates observed in previous studies [10, 32]. As shown in Table 6, for 20 of the leak combinations, we observe supposed password reuse rates above 50%, with one leak combination exhibiting a 98% password reuse rate. These rates far exceed the 9–43% reuse rates previously documented [10, 32]. Thus, we believe these leaks likely contain duplicated data that would not reflect real user password behavior and do not investigate them further.

Finally, three additional leak combinations contain at most a few thousand emails associated with multiple passwords. We require such users for our security evaluation (particularly with studying credential stuffing attacks), so we also filter out these three leak combinations, leaving us with eight Collection#1 leak combinations exhibiting reasonable password reuse rates (those less than 50%) to investigate. Table 6 indicates these eight leaks and their data characteristics. (We caution though that we ultimately lack ground truth on the true nature of these datasets.) Note that we investigate each Collection#1 leak combination separately, rather than aggregate across them.

Duplicate Records. As discussed above, for both the BreachCompilation and Collection#1 datasets, we preserve duplicate (email, password) pairs rather than filter all of them (although we removed entire Collection#1 datasets that exhibited abnormally high rates of data duplication). By preserving these duplicate records which may represent password reuse across breached online services, we can characterize credential stuffing effectiveness when typo-tolerance is not enabled, serving as a baseline to understand the relative impact of enabling typo-tolerance. We note that some of these duplicate records may be due to leak duplication in the datasets, rather than true password reuse by users. As a consequence, our baseline evaluation of credential stuffing's effectiveness without typo-tolerance may be inflated, and our reported results serve as an upper bound on vanilla credential stuffing effectiveness. However, duplicate records *do not* affect our analysis of the absolute impact of enabling typo-tolerance on credential stuffing attacks, and only *lower bounds* our analysis of the impact relative to the baseline (i.e., the relative effectiveness of credential stuffing under typo-tolerance may be even higher than we already report). Thus, preserving duplicate records will not notably affect our findings on the security impact of typo-tolerant password authentication.

4 SECURITY EVALUATION

In this section, we re-evaluate the security implications of typo-tolerant password authentication under a more comprehensive threat model than considered in the existing security analysis by Chatterjee et al. [7]. In particular, the prior analysis considered online password spraying attacks [22], where the attacker guesses common passwords. In this work, we expand the threat model to additionally consider credential stuffing attacks [21, 24], where attackers test a user's leaked password from one online service on other services, and credential tweaking attacks [25], where attackers generate variants of a user's leaked password from one service as login attempts on other services.

Throughout this evaluation, we use the same typo-tolerance password corrector functions as the prior work, as well as the same typo-tolerance policies combining correctors, as discussed in Section 2.1. We first reproduce the prior empirical analysis on our leak datasets (which are more recent and significantly larger than those used previously, although are lacking clear breach provenance), evaluating how typo-tolerance affects password spraying attack success. We then perform our new analysis assessing the influence of typo-tolerance on credential stuffing and tweaking attacks.

4.1 Attack Metrics

To start, we discuss our metrics for attack success under the different password attack models. Both the BreachCompilation dataset and the multiple Collection#1 leak combinations aggregate various leaks, without indicating which data subsets were derived from which leaks. As a result, we cannot group emails and passwords together into individual leaks, preventing a definitive evaluation of attack effectiveness targeting a specific breached online service. Instead, for each separate dataset, we consider the set of passwords associated with each email in that dataset, and compute different attack success metrics to capture the range of potential attack outcomes on the passwords per email. These attack metrics are:

- **Upper Bound:** The upper bound attack metric measures the proportion of emails where *at least one* associated password or password pair would be successfully attacked. This scenario indicates the attacker's best-case success rate, as the attacker successfully attacks every email with *any* susceptible password (or password pair).
- **Lower Bound:** This metric measures the proportion of emails where *all* associated passwords or password pairs would be successfully attacked. This situation represents the attacker's worst-case success rate, where the attacker successfully attacks only the emails where *every* possible password (or password pair) is susceptible.
- **Random:** For this metric, we randomly select one password or password pair for each email, and determine the proportion of emails that would be successfully attacked. This random selection process provides an expected attack success rate.

Essentially, the upper and lower bound metrics characterize the potential range of attack success, whereas the random metric reflects a more realistic attack outcome. Throughout the remainder of this

Dataset	Policy	q = 1000			
		All		Multi	
		Upper	Lower	Upper	Lower
BreachCompilation	C_{None}	6.73	5.25	10.48	1.52
	C_{Top5}	+0.38	+0.31	+0.66	+0.22
C#1: EUcombos	C_{None}	8.39	7.51	10.35	3.32
	C_{Top5}	+0.48	+0.47	+0.70	+0.66
C#1: EUcombos_1	C_{None}	8.96	8.04	11.15	2.48
	C_{Top5}	+0.49	+0.47	+0.83	+0.66
C#1: Gamescombos	C_{None}	8.07	7.12	11.35	2.04
	C_{Top5}	+0.41	+0.38	+0.62	+0.33
C#1: NEW_csp_EUcombo	C_{None}	7.82	6.71	13.21	0.78
	C_{Top5}	+0.33	+0.29	+0.64	+0.19
C#1: OC_BTCcombos	C_{None}	5.66	3.90	11.98	4.26
	C_{Top5}	+0.35	+0.28	+0.58	+0.27
C#1: OC_Porncombos	C_{None}	11.68	10.70	19.85	2.89
	C_{Top5}	+0.48	+0.45	+0.94	+0.44
C#1: OC_UKcombos	C_{None}	9.95	8.09	16.72	2.06
	C_{Top5}	+0.72	+0.63	+1.24	+0.58
C#1: RUcombo	C_{None}	13.55	6.69	36.75	2.20
	C_{Top5}	+0.49	+0.33	+0.88	+0.09

Table 1: The effectiveness of password spraying when password typo-tolerance is disabled (C_{None}) compared with using the C_{Top5} typo-tolerance policy, across different datasets for q = 1000 attack queries. For each leak, we evaluate attack success on all emails (labeled as *All*) and only emails with multiple passwords (labeled as *Multi*), using the upper and lower bound attack success metrics. For ease of comparison, the C_{Top5} attack metrics are the percentage point increases/deltas (indicated by the + sign) in password spraying success over C_{None}, rather than the total attack success rate.

section, we use these three attack success metrics to characterize the effectiveness of our different attack models.

In addition, throughout our analysis, we will often use percentage points (*pp*) for comparing two percentages, where the percentage points show the arithmetic difference between two percentages (e.g., increasing from 50% to 55% is a 5*pp* increase). We use this unit as it represents the absolute difference between two percentages rather than a relative one, which is particularly useful when interpreting the difference between two attack success rates or population percentages.

4.2 Password Spraying Attacks (Replication)

In the prior security analysis of typo-tolerant password authentication, Chatterjee et al. [7] used password leaks from three websites (RockYou, phpBB, and Myspace) to empirically demonstrate that typo-tolerance did not significantly exacerbate password spraying attacks, where attackers guessed common passwords. These three leaks are dated (circa 2009) and small (only the Myspace leak exceeded 1M users), compared to our more recent and significantly larger datasets. Here, we replicate their analysis on our distinct password datasets to confirm the prior work's observations.

Analysis Method. We evaluate two typo-tolerance policy configurations: no typo-tolerance (C_{None}) and the most permissive typo-tolerance policy (C_{Top5}). (Recall that Section 2.1 discussed the nature of these policies.) For our password leak datasets, some emails are associated with multiple passwords while others are tied to only one. While analyzing all emails provides the most comprehensive analysis of password spraying effectiveness, our subsequent analyses of credential stuffing and tweaking attacks (in Sections 4.3 and 4.4) are restricted to only emails with multiple passwords. Thus, to support more direct comparisons across attacks, we consider both populations separately (all emails and only multi-password ones). Finally, password spraying attacks depend on a parameter q, indicating the number of attack queries attempted. We evaluate q = 10, 100, 1000, as also done by Chatterjee et al. [7]. We note that typo-tolerance only affects online attacks, and websites often deploy defenses (e.g., rate limiting, blocklisting) that limit the number of queries that attackers can reasonably make (i.e., q = 1000 is less feasible in practice).

A password spraying attack succeeds against an (email, password) pair if the attacker is able to successfully guess the password within q queries. The optimal guessing strategy is for the attacker to attempt passwords in order of decreasing popularity, assuming the attacker knows the password distribution. While attackers typically lack this knowledge in practice, in our analysis, we assume the attacker knows the target dataset's password distribution and we simulate an optimal attack. By doing so, we model the best-case scenario for the attacker.

Analysis Results. Table 1 displays the password spraying attack results across different target datasets and typo-tolerance policies, for attacks consisting of q = 1000 queries, the strongest password spraying attack we evaluate. We list the upper bound and lower bound attack success rates (we elide the random metric for space). In Table 7 of the Appendix, we show the password spraying attack results for q = 10 and 100, the weaker attacks.

Without typo-tolerance (where the policy is C_{None}), we observe varying password spraying effectiveness across the different password datasets and email populations. However, when enabling the most permissive typo-tolerance policy (C_{Top5}), we see that the increase in password spraying success is minimal across all settings, even though we are considering our largest attack size (q = 1000). The upper bound increase in attack success is below 1 percentage point (*pp*) in all cases except one (targeting multi-password emails in Collection#1's *OC_UKcombos* leak, with an attack success increase of 1.24*pp*). For weaker attacks (q = 10 and 100), typo-tolerance has an even smaller impact on attack success.

Thus, we replicate the same conclusion as prior work [7], that typo-tolerant password authentication does not significantly advantage password spraying attacks.

4.3 Credential Stuffing Attacks

Here, we assess how typo-tolerant password authentication is affected by credential stuffing attacks. This class of attacks was not considered in the prior security analysis of typo-tolerance [7], thus our security evaluation encompasses a broader threat model.

Analysis Method. Credential stuffing attacks involve two passwords from a user, one *leaked password* (from one online service) that the attacker has access to and one *targeted password* (on another online service) that the attacker aims to guess. To evaluate credential stuffing attacks using the leak datasets, we only consider emails associated with multiple passwords within the same dataset. We analyze all *ordered* password pairs per email, where each pair

Dataset	C_{None}		C_{Top1}		C_{Top2}	
	Upper / Lower	Random	Upper / Lower	Random	Upper / Lower	Random
Breach Compilation	20.9 / 16.2	17.1	+1.3 / +0.6	+0.7	+3.4 / +1.7	+2.0
C#1: Eucombos	49.8 / 41.6	44.1	+1.4 / +0.9	+1.1	+7.9 / +6.2	+6.3
C#1: EUCombos_1	30.7 / 26.8	27.9	+1.9 / +1.3	+1.5	+4.4 / +3.5	+4.2
C#1: Gamescombos	30.1 / 28.2	28.6	+10.3 / +0.5	+1.8	+14.2 / +3.2	+4.7
C#1: NEW_csp_EUcombo	~0.0 / ~0.0	~0.0	+3.1 / +0.7	+1.1	+5.2 / +2.0	+2.5
C#1: OC_BTCcombos	47.6 / 41.8	43.2	+0.2 / +0.1	+0.1	+0.9 / +0.3	+0.5
C#1: OC_Porncombos	11.3 / 8.2	8.9	+1.7 / +1.2	+1.3	+2.7 / +2.0	+2.2
C#1: OC_UKcombos	11.8 / 9.1	9.8	+1 / +0.5	+0.6	+4.1 / +2.4	+2.8
C#1: RUcombo	7.4 / 6.0	6.3	+6.8 / +0.2	+0.3	+7.3 / +0.4	+0.5

Dataset	C_{Top3}		C_{Top4}		C_{Top5}	
	Upper	Random	Upper	Random	Upper	Random
Breach Compilation	+8.1	+4.0	+8.9	+4.3	+9.0	+4.4
C#1: Eucombos	+10.8	+8.6	+11.1	+8.8	+11.2	+8.8
C#1: EUCombos_1	+12.0	+8.0	+12.4	+8.1	+12.5	+8.2
C#1: Gamescombos	+22.8	+7.3	+23.4	+7.6	+23.5	+7.6
C#1: NEW_csp_EUcombo	+14.1	+5.4	+15.0	+5.8	+15.3	+5.8
C#1: OC_BTCcombos	+5.5	+2.5	+6.0	+2.7	+6.0	+2.7
C#1: OC_Porncombos	+7.9	+4.2	+8.3	+4.3	+8.4	+4.4
C#1: OC_UKcombos	+14	+6.6	+14.4	+6.8	+15.0	+6.8
C#1: RUcombo	+28.6	+1.3	+28.8	+1.3	+33.4	+1.5

Table 2: The effectiveness of credential stuffing attacks when typo-tolerance is disabled (C_{None}) compared with using the C_{Top1} through C_{Top5} typo-tolerance policies, across different leak datasets. For each leak, we evaluate attack success at the email granularity, using the upper bound, lower bound, and random attack success metrics. For C_{Top3} to C_{Top5}, we elide the lower bound values as they do not increase compared to C_{Top2}, because the additional correctors applied in those policies are non-symmetric. For ease of comparison, the C_{TopX} attack success metrics are the percentage point increases/deltas (as indicated by the + sign) in credential stuffing success over C_{None}, rather than the total attack success rate.

represents a (leaked, targeted) password pair. For a given password pair, the credential stuffing attack succeeds if the leaked password matches, or is corrected to if applying typo-tolerance, the targeted password. Considering all (leaked, targeted) password pairs per email, we assess the success rate of credential stuffing by comparing all five typo-tolerance policies and no typo-tolerance, using the upper bound, lower bound, and random attack success metrics.

Note that the pair ordering is important (i.e., identifying which password is the leaked one versus the targeted one) because when we consider typo-tolerance, the *rm_first*, *rm_last*, and *n2s_last* password correctors (as discussed in Section 2.1) are not symmetric functions. For example, `password1` corrects to `password` under the *rm_last* corrector, but not vice versa. For any email with a password pair where credential stuffing succeeds only when applying a non-symmetric corrector, that email contributes to the upper bound attack success rate *but not* to the lower bound rate, as the inverted pair would not exhibit a successful stuffing attack (as the corrector is non-symmetric). As a consequence, the lower bound attack metric does not increase for policies more permissive than C_{Top2} (i.e., C_{Top3} through C_{Top5}), as these policies only incorporate additional non-symmetric correctors.

Analysis Results. In Table 2, we show the effectiveness of credential stuffing attacks across different leak datasets and different typo-tolerance policies. We list the upper bound, lower bound, and random attack success metrics, showing the percentage point increases in attack success when a typo-tolerant policy is enabled, compared to no typo-tolerance.

When typo-tolerance is not enabled (C_{None}), we find that credential stuffing attacks are successful at targeting a large portion of emails, with the random attack success metric primarily ranging from 6.3% to 44.1% of emails. The exception is the Collection#1 *NEW_csp_EUcombo* leak, where only a small but non-zero number (less than 0.1%) of emails exhibited password reuse. As we do not know the origin of this leak, it is unclear why the password reuse rate is so low, although it is possible that some (but not all) password reuse cases were deduplicated. The observed credential stuffing success rates are commensurate with prior studies [10, 32] that analyzed the password reuse rates between distinct website password dumps, which found reuse rates mostly between 9-20% but rising as high as 43%. The high success rates also reflect the reality that credential stuffing attacks have been damaging in practice [1, 24, 30].

Once enabling the simplest and most limited typo-tolerance policy (C_{Top1}), we observe an increase in the random credential stuffing success rate by up to 1.8*pp*, which already exceeds the largest increase in upper bound password spraying success rate seen in Table 7 (1.24*pp*, when using the C_{Top5} typo-tolerance policy under password spraying attacks with 1000 guesses). The upper bound metrics reflect credential stuffing success increases by 1*pp* or more for all leaks except one (Collection#1's *OC_BTCcombos*, with an upper bound increase of 0.2*pp*). Thus, even with the most restrictive typo-tolerance password policy, we already see a noticeable impact from typo-tolerance on credential stuffing attacks.

As we expand the typo-tolerance policy, the attack metrics increase, with the largest increases occurring when expanding from C_{Top1} to C_{Top2}, and C_{Top2} to C_{Top3}. With the most tolerant policy (C_{Top5}), we find that credential stuffing is significantly more successful than without typo-tolerance. Under the random attack success

metric, enabling C_{Top5} allows the credential stuffing success rate to increase by 1.5–8.8*pp*, with the majority of leak datasets exhibiting a delta of over 5*pp*. In the worst-case scenario for the typo-tolerant system, the upper bound credential stuffing success rate increases by between 6–33.4*pp*. Thus, a significant portion of users become newly susceptible to credential stuffing attacks once typo-tolerance is enabled. Furthermore, the relative strength of credential stuffing increases notably through enabling typo-tolerance. For example, under the random attack success metric with C_{Top3}, credential stuffing is 23% more effective in the BreachCompilation dataset and 67% more effective in the Collection#1 *OC_UKcombos* dataset.

Our evaluation here demonstrates empirically that typo-tolerance *can* actually severely reduce password security in practice, by empowering credential stuffing attacks. The underlying issue is that users often use minor variations of a password across online services [10], and these variations are closely related to the type of transformations applied by typo-tolerance correctors. Thus, during a credential stuffing attack, a user's leaked password often can be corrected to their password on a target service. Our findings counter the initial security analysis of typo-tolerant password authentication [7], which concluded that such systems could increase authentication usability with negligible security degradation. To be clear though, we do not find an error with the prior work's analysis. Rather, the existing analysis only considered a threat model of password spraying attacks, whereas in our evaluation, we expand the threat model to include credential stuffing, a realistic inclusion given prominent credential stuffing attacks in the wild [1, 24, 30].

4.4 Credential Tweaking Attacks

With credential stuffing, an attacker only attempts one guess at a target password (i.e., the leaked password). However, as observed in prior work [10, 25] as well as our credential stuffing attack analysis in Section 4.3, users often use similar but distinct passwords across websites, frequently applying several common categories of variations. As a consequence, an attacker could apply similar variations to a leaked password in order to generate additional fruitful login attempts (in addition to testing the leaked password itself). Prior work has demonstrated that such *credential tweaking attacks* can be significantly stronger than credential stuffing. Here, we explore the degree to which typo-tolerant password authentication schemes exacerbate credential tweaking attacks, compared to credential stuffing attacks (as analyzed in Section 4.3).

Analysis Method. For this analysis, we rely on a state-of-the-art credential tweaking model developed by Pal et al. [25]. This model, which is open-sourced [3], takes a leaked password as an input and generates password variations for further login attempts.

Our attack evaluation is similar to that of credential stuffing (as discussed in Section 4.3). However, given a (leaked, targeted) password pair, we do not only check if the leaked password matches or is corrected to the targeted password (simulating a successful credential stuffing attack). Instead, we also apply the credential tweaking model to the leaked password to generate nine additional unique password variations (for a total attack strength of 10 guesses, the weakest attack evaluated by Pal et al. [25])[1]. We consider the credential tweaking attack successful if the leaked password or any of its nine variants (for a total of 10 password guesses) match or are corrected to the targeted password. As before, we compute the upper bound, lower bound, and random attack success rates.

[1]We ignore the 21 emails (out of 100K) with a password where the model could not generate nine unique variants.

Policy	Upper	Random	Lower
C_{None}	37.8	26.1	20.1
C_{Top1}	+5.0	+2.3	+0.7
C_{Top2}	+6.7	+3.7	+1.9
C_{Top3}	+13.9	+6.8	+2.3
C_{Top4}	+15.0	+7.3	+2.3
C_{Top5}	+15.1	+7.4	+2.3

Table 3: The effectiveness of credential tweaking attacks when typo-tolerance is disabled (C_{None}) compared with using the C_{Top1} through C_{Top5} typo-tolerance policies. Due to computational constraints, we use a randomly selected set of 100K multi-password emails from the BreachCompilation dataset. We evaluate the upper bound, lower bound, and random attack success metrics. For ease of comparison, the C_{TopX} values are the percentage point increases/deltas (as indicated by the + sign) in credential tweaking success over C_{None}, rather than the total attack success rate.

The credential tweaking model is computationally intensive and time consuming though, as documented by the model authors [25]. Hence, we limit our evaluation to a randomly chosen set of 100K emails associated with multiple passwords from the BreachCompilation dataset[2]. We note that Pal et al. [25] analyzed their model using a random sample of the same size on the same source dataset, thus we are adhering to the same analysis parameters.

Analysis Results. Table 3 shows the upper bound, lower bound, and random attack success rates for credential tweaking attacks across different typo-tolerance password policies (including no typo-tolerance). Comparing credential tweaking and credential stuffing attacks (in Table 2) without typo-tolerance (C_{None}), we observe that credential tweaking is significantly more effective, as expected. With the random attack success metric, credential tweaking succeeds with 26.1% of emails, compared to 17.1% for credential stuffing. This 9.0*pp* difference is commensurate with the findings by Pal et al. [25], who observed a 9.9*pp* delta between the two attacks during their evaluation[3]. In the best-case scenario for the attacker, over a third of emails (37.8%) can be successfully attacked; in the worst-case scenario for the attacker, 20.1% of emails are still susceptible.

When enabling typo-tolerance, we observe that credential tweaking becomes notably more effective. Using the random attack success metric, credential tweaking's success rate increases by 6.8*pp* under the C_{Top3} policy, up to 7.4*pp* under the C_{Top5} policy (a 28% gain). Even enabling the least permissive typo-tolerance policy

[2]Even for our small sample of emails, it took four hours to generate the credential tweaking model's attack guesses for our evaluation, using 20 cores on a server with 256 GB of memory.

[3]We note that the credential tweaking analysis by Pal et al. [25] is not identical to ours, making a direct comparison inexact. Our analysis simulates both credential stuffing and tweaking attacks, and compares their effectiveness. In contrast, the prior evaluation removed password reuse cases (essentially eliminating the effectiveness of credential stuffing), and analyzed the effectiveness of credential tweaking. However, as both analyses isolate the impact of credential tweaking from that of credential stuffing, we believe the comparison remains fair and meaningful.

C_{Top1} still results in a 2.3*pp* increase in attack success. In the upper bound case (the best-case scenario for the attacker), credential tweaking is able to successfully attack an additional 15.1% of users under C_{Top5}, a 40% gain in attack potency compared to C_{None}.

Our findings here reinforce the conclusion from our credential stuffing attack analysis (Section 4.3). We again empirically observe that typo-tolerance can result in consequential degradation in password security, once considering a broader attack model than that of the prior security analysis of typo-tolerant password authentication systems [7]. We also note that our evaluation is of the weakest credential tweaking attack considered by Pal et al. [25], who investigated tweaking attacks consisting of between 10 to 1000 total guesses, finding that using more generated password guesses resulted in more effective tweaking attacks. Thus, our findings on the impact of typo-tolerance on credential tweaking attacks can be viewed as lower bounds, meaning tweaking attacks with a higher number of guesses should be even more effective with typo-tolerance enabled.

5 HARDENING TYPO-TOLERANT PASSWORD AUTHENTICATION

In Section 4, we evaluated typo-tolerant password authentication under a more comprehensive threat model than considered in the existing security analysis [7]. We identified that, counter to the prior findings, typo-tolerance can result in notable security degradation in practice, as credential stuffing and tweaking attacks are significantly more effective when typo-tolerance is enabled. We concluded that typo-tolerance increases password usability (as demonstrated by Chatterjee et al. [7]) but at a security cost.

In this section, we investigate whether we can reduce typo-tolerance's security loss while maintaining its functionality. In particular, we aim to predict if a given password is newly susceptible to credential stuffing once typo-tolerance is enabled. We will call such affected passwords *susceptible passwords*. With such a capability, an online service could identify and selectively disable typo-tolerance for users predicted to be negatively impacted by the authentication scheme (i.e., using a susceptible password), while preserving typo-tolerance for other users.

We first consider using existing techniques that assess password security, specifically password strength meters (PSMs). Our hypothesis is that weak passwords correlate with susceptible ones, as perhaps users who use minor variants of a password across sites may more likely choose weaker passwords. In Section 5.1, we evaluate whether we can use the popular and open-source `zxcvbn` PSM [12] to accurately predict password susceptibility.

However, identifying weak passwords is an inherently different task from determining password susceptibility, a notion that will be reinforced by our PSM evaluation results in Section 5.1. As an alternative, we explore a machine learning approach to develop a binary classifier for susceptible passwords. Given the specific transformations performed by typo-tolerance correctors, our intuition is that susceptible passwords exhibit specific structure which a classifier could potentially identify. In Section 5.2, we discuss how we design such a classifier, and in Section 5.3, we evaluate its performance, demonstrating practical value.

Policy	Strength Score	Users w/ Weak Password	FPR	FNR
C_{Top1}	<2	0.73	0.73	0.27
	<3	0.90	0.90	0.10
C_{Top2}	<2	0.73	0.73	0.26
	<3	0.90	0.90	0.09
C_{Top3}	<2	0.73	0.73	0.19
	<3	0.90	0.90	0.06
C_{Top4}	<2	0.73	0.72	0.19
	<3	0.90	0.90	0.06
C_{Top5}	<2	0.73	0.72	0.19
	<3	0.90	0.90	0.06

Table 4: The effectiveness of disabling typo-tolerance for users of weak passwords. On the BreachCompilation dataset, we evaluate the proportion of users with weak passwords for different weakness thresholds, representing the population with typo-tolerance disabled. Across the typo-tolerance policies, we also evaluate credential stuffing attack success and the proportions of users for whom typo-tolerance should have been disabled (false negative rate, or FNR) or should not have been disabled (false positive rate, or FPR).

Note that here we explore hardening typo-tolerance against credential stuffing attacks, and we leave the investigation of credential tweaking attacks for future work. This decision is in part driven by the high computational cost of the state-of-the-art credential tweaking attack model [25] (used in Section 4.4), which limits the amount of training and testing data we can practically use for our machine learning development. Furthermore, credential stuffing attacks are commonplace in practice [24, 30, 30], whereas to our knowledge, credential tweaking attacks remain largely theoretical.

5.1 Disabling Typo-Tolerance for Weak Passwords

We first explore using a PSM (specifically, `zxcvbn` [12]) to classify susceptible passwords as weak ones. We evaluate the impact of disabling typo-tolerance for weak passwords on users and credential stuffing attacks.

Analysis Method. Our evaluation here mirrors the credential stuffing attack analysis from Section 4, using the BreachCompilation dataset. The only difference is that in this analysis, we do not permit typo-tolerance for user passwords that are classified as weak by `zxcvbn`. Given a password, `zxcvbn` outputs a strength score ranging from 0 (weakest/too guessable) to 4 (strongest/very unguessable). We consider two different thresholds for defining weak passwords: 1) strength score is less than 2 (too or very guessable) and 2) strength score is less than 3 (somewhat guessable or weaker).

Across the different typo-tolerance policies and weak password thresholds, we measure the population of users where typo-tolerance is disabled due to the use of a weak password. We then evaluate credential stuffing attacks under the random attack success metric (where for each user, one password is randomly selected as the leaked password, and another as the targeted password). We define a false positive as a case where typo-tolerance is disabled for a user but they are not using a susceptible password (i.e., credential stuffing would not have succeeded even if typo-tolerance was enabled). A false negative is when credential stuffing succeeds for a user due to typo-tolerance remaining enabled. We characterize the

false positive and false negative rates to identify the effectiveness of using a PSM for selectively disabling typo-tolerance.

Analysis Results. Table 4 depicts the impact of disabling typo-tolerance for weak passwords. We find that such actions do provide security benefits. False negatives indicate users of susceptible passwords who should have typo-tolerance disabled. Across all typo-tolerance policies and weak password thresholds, we observe false negative rates (FNR) between 6-28%, indicating that the majority of users negatively impacted by typo-tolerance do have it disabled.

However, we observe that weak passwords are used by 73-90% of users, depending on the weak password threshold. These proportions are commensurate with findings from prior empirical analysis of PSM scoring [11]. As a result, typo-tolerance is disabled for most users, even when it need not be in most cases (as the false positive rate is high). Thus, disabling typo-tolerance for weak password users does not preserve the functionality of typo-tolerance.

We conclude that using a PSM to identify weak passwords and selectively disable typo-tolerance for them is not a viable method for hardening typo-tolerance. As an alternative approach, we next explore developing a machine learning model specifically trained to identify susceptible passwords. Intuitively, PSMs were not developed for such a task, so a model specifically designed for the problem domain should outperform it (in addition, such a model could use password strength as a feature while leveraging additional features for improved accuracy).

5.2 Machine Learning Model Design

Model Features. As our model classifies password inputs, we derive features from password characteristics. The full set of 46 features considered are summarized in Table 8 of the Appendix, and consists of categorical, numerical, and Boolean features. At a high level, these features can be divided into several groups:

- One set of features focuses on capturing the password composition, structure, and complexity, including the classes of characters used (i.e., uppercase and lowercase letters, digits, and symbols), the frequency of transitions between character classes, password entropy, and password strength. Our rationale for using such features is that there may be a correlation between the password structure and complexity, and the likelihood that the password is used in slightly modified forms across online services.
- Another group of features cater to the typo-tolerance correctors, and characterize the characters involved in corrector transformations. For example, the *n2s_last* corrector operates on the last password character, changing it to the equivalent character under the shift-key modifier. We use a feature indicating if the last character would be changed by the corrector (e.g., "1" would be transformed to "!", but "T" is already capitalized and would not be modified). Similarly, to account for the *swc_first*, *rm_first*, and *rm_last* correctors, the features also capture the character types of the first and last password characters, as well as their similarities to adjacent characters. Using such corrector-specific features, we aim for the model to identify when a password may be transformed by a corrector (if not, then the password's security is not affected by the corrector).
- We also consider password popularity, as minor variants of popular passwords may be commonly used across sites.

Password Labels. A given password may be used by many users, and its use may be susceptible under some users and not others. In other words, given a password, some users may vary it slightly for use across online services, while other users will not. As a consequence, training a machine learning model on all instances of a password's use (across users) would result in a noisy signal, where the same password input will have different labels depending on the instance. To avoid unclear decision boundaries, we train our model on distinct passwords, providing a consistent label for each password input.

This approach raises the question of what the password label should be, given some instances of the password will be susceptible and others will not. We explore different password labels depending on whether the proportion of the password's users that are susceptible exceeds a threshold, which we call the *label threshold*. Using a 0% label threshold prioritizes security, as a distinct password is classified as susceptible if any of its instances are susceptible. Using a higher label threshold trades security for functionality, as the classifier avoids labeling a distinct password as susceptible if only a fraction of its users are susceptible. We evaluate our models on data labeled using 0%, 10%, and 25% label thresholds. Note that password labels also depend on the typo-tolerance policy, as the policy dictates which password instances are susceptible.

Classification Model. We train machine learning models for each of the five individual typo-tolerance correctors, as well as for each of the five typo-tolerance policies (C_{Top1} to C_{Top5}). We note that one could implement a policy model by using multiple corrector models; however, our subsequent exploration did not identify classification accuracy benefits to doing so, while using multiple models results in additional design complexity and performance costs. Examining the individual corrector models does provide us with insights on the characteristics of passwords susceptible under each corrector function.

We use a binary decision tree for our classification model. While we initially explored several alternative designs (including logistic regression, support vector machines, random forests, and neural networks), we did not observe superior classification performance. Decision trees also are directly interpretable, and we explore the most influential model features in Section 5.3.1.

We evaluate our model on the BreachCompilation leak dataset, considering emails with multiple passwords (as we are defending against credential stuffing attacks). We first randomly divide the dataset's multi-password emails into two groups, where one group consists of 90% of the emails and is used as a training evaluation dataset, and the remaining 10% of emails are reserved as a holdout test dataset (which we will use for evaluating classifier performance on password instances, rather than only considering distinct passwords). For the training evaluation dataset, we generate labels for distinct passwords based on the typo-tolerance policy and the label threshold. This data exhibits heavy class imbalance, where the majority of distinct passwords are not susceptible (the exact ratio varies depending on the labeling). We downsample the dominant class (not susceptible passwords) for a balanced training dataset[4] (although we explored using imbalanced data, observing slightly

[4]Note that even after downsampling, our training datasets are sizable, with the smallest training dataset containing 300K unique passwords.

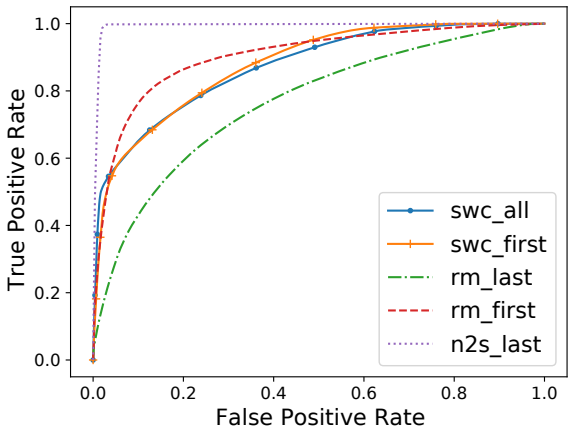

Figure 1: ROC curves for the individual corrector models trained on data labeled using a 0% label threshold.

worse model performance). Finally, we train our decision tree models on the training dataset using 10-fold cross-validation[5], using grid search for hyperparameter tuning (selecting a maximum tree depth of 15). In Section 5.3, we discuss our training results as well as our model performance on the holdout test dataset, where we can evaluate model performance on password instances in addition to distinct passwords.

5.3 Machine Learning Model Evaluation

In this section, we evaluate our password classifiers, presenting both the model performance during training (using cross-validation), as well as on a holdout test dataset that simulates the realistic application of the classifiers. We also characterize the computational costs of training and deploying the model.

5.3.1 Training Results. Here, we evaluate the results from training our different classifier models. As with any machine learning approach, we aim to maximize recall while minimizing false positives. However, the cost of false positives in our scenario may be considered relatively low, as a false positive results in disabling typo-tolerance for a user. When this occurs, the user loses the usability benefits of typo-tolerance but does not suffer security consequences. Thus, a model with high recall but a modest false positive rate (FPR) can still be useful in practice, offering a different functionality-security tradeoff for typo-tolerance compared to fully enabling or disabling typo-tolerance.

Corrector Models. Figure 1 depicts the receiver operating characteristic (ROC) curves for the five individual corrector models, where the data is labeled using the 0% label threshold, and each curve is averaged across the cross-validation folds. The *n2s_last* model has the highest ROC curve, with 99% recall at a 2% FPR. The other corrector models offer less favorable recall/FPR tradeoffs, with the *rm_last* model performing worst. Achieving 75% recall with the *rm_last* model results in a 36% FPR, whereas for the *swc_all*, *swc_first*, and *rm_first* models, the same recall level results in FPRs of 20%, 20%, and 10%, respectively. We observe the same patterns in the recall/FPR tradeoffs when the corrector models are trained on data labeled with the 10% and 25% label thresholds (as seen in the ROC curves of Figures 3 and 4, in the Appendix).

The performance of these models demonstrates that passwords do exhibit characteristics that indicate likely susceptibility, although the signal is (unsurprisingly) noisy. Except for the *n2s_last* corrector model, we are unable to obtain high recall (e.g., >95%) at low FPRs (e.g., <5%). However, we can still achieve meaningful recall levels (e.g., 75%) at modest FPRs (e.g., 10-40%). Such operating points would protect the majority of users from the security degradation introduced by a typo-tolerance corrector, while still maintaining typo-tolerance for most users. Thus, we argue that these models can be useful in practice as they offer a different security-functionality tradeoff compared to disabling typo-tolerance (resulting in no usability gains but no security losses) and fully enabling typo-tolerance (affording the full usability benefits with the full security degradation costs observed in Section 4).

Top Features. We inspect the most influential features in the corrector decision-tree models to gain insights into their inner workings, using the feature importance scores. In particular, the dominant features represent password characteristics that most signal susceptibility (or lack thereof) under a specific typo-tolerance corrector, which also reflect the types of modifications that users apply when using similar passwords across online services. Notably, password strength is not a top feature for any corrector, aligning with our prior finding that a PSM would serve as a poor classifier for susceptible passwords.

- *swc_all*: The most important features for the *swc_all* models are whether the password consists of all uppercase letters, whether the password contains only uppercase letters and digits, and the password length. The most susceptible passwords are those shorter than 13 characters with only uppercase letters (and possibly digits).
- *swc_first*: The influential features for the *swc_first* models are whether the first character is an uppercase or lowercase letter, and whether the first two characters are from the same character class. Susceptible passwords tend to be those with both uppercase and lowercase letters as the first character and a lowercase letter as the second character.
- *rm_last*: The top features for the *rm_last* models are not as discriminative as with the other corrector models (unsurprising as the *rm_last* models provided the worst recall/FPR tradeoffs). The most impactful feature is the password length, with the next two highest ranked features being the number of character class transitions in the password and the length of the longest single-character-class substring. Passwords shorter than 17 characters with few character class transitions are more likely to be susceptible under the *rm_last* corrector.
- *rm_first*: The dominant feature for the *rm_first* models is whether the password consists only of digits. The password length and popularity are two other influential features. The most susceptible passwords under the *rm_first* corrector are popular all-digit ones with lengths less than 16.

[5] As our training data consists of distinct passwords, during cross-validation, the set of passwords our model is trained on has no overlap with the set of passwords it is tested on. Thus, our model is evaluated on passwords it did not observe during training.

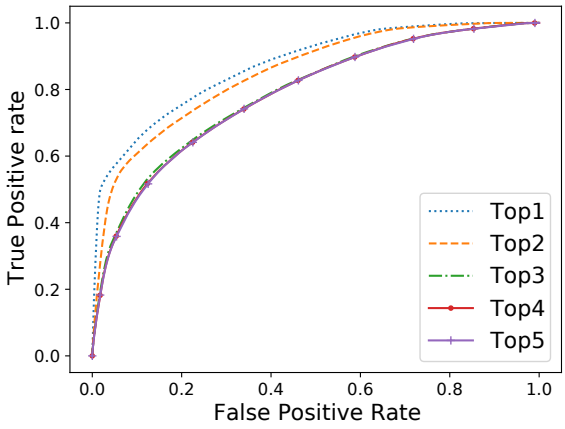

Figure 2: ROC curves for the policy models trained on data labeled using a 0% label threshold.

- *n2s_last*: The notable features for the *n2s_last* models are whether the last character can be shift-key modified, whether the password's last two characters are the same character class, and whether the password consists of lowercase letters and digits. Susceptible passwords are often those with an uppercase letter or symbol as the last character (i.e., a character that is the shift-key modification of another character), where the remainder of the password consists of lowercase letters and digits.

Policy Models. Figure 2 shows the ROC curves for the five policy models where each curve is averaged across the cross-validation folds. Here, the models are trained on data labeled using a 0% label threshold. We observe that each policy model's performance is similar to that of the worst-performing corrector in the policy. C_{Top1} consists only of the *swc_all* corrector, so their models' ROC curves are identical. The ROC curves for the C_{Top1} and C_{Top2} models are similar. As C_{Top2} adds the *swc_first* corrector, this similarity aligns with the similar ROC curves for the *swc_all* and *swc_first* corrector models. Finally, the policy models for C_{Top3}, C_{Top4}, and C_{Top5} (all of which include the *rm_last* corrector) present nearly identical performance, matching the performance of the *rm_last* corrector model (the worst-performing corrector model). Achieving 75% recall with the C_{Top3}, C_{Top4}, and C_{Top5} policy models result in a 36–38% FPR. Meanwhile, the same recall level for the C_{Top1} and C_{Top2} policy models result in a FPR of 21% and 26%, respectively. As with the corrector models, we observe the same patterns for policy models trained on data labeled using the 10% and 25% label thresholds (as visible in Figures 5 and 6 in the Appendix).

These results show that to a modest extent, we can discriminate between susceptible and non-susceptible passwords under a typo-tolerance policy. Our policy models exhibit performance that is largely bounded by the performance of the worst-performing model for a corrector in the policy. As with the corrector models, we failed to achieve high recall at a low FPR. However, our policy models still provide meaningful recall levels with modest FPRs, allowing a typo-tolerant authentication system to trade off between functionality and security. (We note that these policy models also offer better security-functionality tradeoffs than with using a password strength meter as a classifier, as evaluated in Section 5.1.)

5.3.2 Holdout Test Set Results. Recall from Section 5.2 that our password classification task is defined on distinct passwords rather than password instances. This formulation allowed us to pursue models that leverage password characteristics to predict likely susceptibility (whereas models operating on password instances would encounter the same password input with varying output labels, as some instances are susceptible while others are not). Section 5.3.1's evaluation of our corrector and policy password classifiers similarly considered model performance across distinct passwords (in the training dataset). However, in practice, policy models would be applied to password instances across users. In this section, we investigate how the classifiers perform in this realistic setting, using our holdout test set of emails (the random 10% of emails withheld from the training evaluation, as discussed in Section 5.2).

Analysis Method. For the five typo-tolerance policies, we use the same policy model parameters as evaluated in Section 5.3.1, except with the models trained on distinct passwords in the entire training dataset (as our earlier evaluation used cross-validation). We again consider the 0%, 10%, and 25% label thresholds for labeling the training data. For each policy model, we consider multiple recall operating points, exploring the tradeoff between recall and the false positive rate. Specifically, we consider models tuned to 10%, 25%, 50%, 75%, and 90% recall. We then apply these models to classify a randomly selected password for each email in the holdout test dataset, simulating the use of our models at an online service.

For each model configuration (training label threshold, typo-tolerance policy, and recall operating point), we analyze the proportion of emails with their (randomly selected) password classified as susceptible (to credential stuffing once typo-tolerance is enabled). These would be emails where typo-tolerance (using the model's corresponding policy) would be disabled, representing the total user-facing impact of applying the model. To tease apart the security and functionality implications of the models, we assess whether the passwords are actually susceptible or not, in a similar fashion as done for the random attack success metric in Section 4.3. For each email, we randomly select a second password as the leaked password. We identify whether the typo-tolerance policy corrects the leaked password to match the original randomly selected password (i.e., the original password is susceptible in reality), and whether our model predicts the original password as susceptible. If the model predicts susceptibility but the email's password is not susceptible in reality, we consider the email as a false positive. Here, our model would cause typo-tolerance to be unnecessarily disabled for the email. Similarly, if the model does not predict susceptibility but the email's password is actually susceptible, we consider the email as a false negative.

Analysis Results. Table 5 shows the model susceptibility prediction rate, the prediction false positive rate (FPR), and the prediction false negative rate (FNR) for our various policy models when trained on data using the 10% label threshold. Across all five policies, we observe recall operating points that offer modest FPRs and FNRs. For example, at 50% recall, the five policy models predict between 24–41% of emails to have a susceptible password, with FPRs ranging between 23–39% and FNRs ranging from 23–35%.

Label Threshold	Policy	Recall	Susceptible	FPR	FNR
10%	Top1	10%	0.19	0.18	0.53
		25%	0.19	0.19	0.46
		50%	0.25	0.35	0.32
		75%	0.58	0.58	0.16
		90%	0.70	0.70	0.11
	Top2	10%	0.21	0.20	0.56
		25%	0.22	0.21	0.47
		50%	0.24	0.23	0.35
		75%	0.62	0.61	0.12
		90%	0.72	0.71	0.05
	Top3	10%	0.27	0.25	0.37
		25%	0.28	0.25	0.33
		50%	0.41	0.39	0.24
		75%	0.64	0.62	0.14
		90%	0.74	0.73	0.09
	Top4	10%	0.27	0.25	0.38
		25%	0.28	0.25	0.32
		50%	0.40	0.37	0.23
		75%	0.63	0.61	0.13
		90%	0.73	0.71	0.09
	Top5	10%	0.23	0.20	0.42
		25%	0.25	0.22	0.37
		50%	0.39	0.36	0.24
		75%	0.59	0.57	0.14
		90%	0.69	0.68	0.09

Table 5: The performance of the password classifier models on the holdout test dataset, trained on data labeled using a 10% label threshold. We evaluate the five policy models each tuned to varying recall operating points, and determine the proportion of emails whose randomly selected password is flagged as susceptible by our models, as well as the models' false positive and false negative rates.

In practice, deploying such models would result in typo-tolerance being disabled for a quarter or a third of users, while eliminating the security degradation for up to three-quarters of the users who would be negatively affected by typo-tolerance otherwise. For many online services, this may be a suitable tradeoff, limiting the security degradation from typo-tolerance while maintaining the majority of its functionality.

In Table 9 in the Appendix, we show the equivalent model evaluation results for models where the training data label threshold is 0%. We observe that the model susceptibility prediction rate is above 63% for all policies and recall operating points, with similarly high FPRs. This outcome is expected though, as using a 0% label threshold aims to maximize security, and any distinct password that had a susceptible instance during training (even if rare) was labeled as susceptible. We do note that there are settings where the FNR is low yet a non-trivial fraction of emails would not have typo-tolerance disabled. For example, the C_{Top2} model operating at 90% recall exhibits a 3% FNR while predicting susceptibility for 78% of emails. This result demonstrates that typo-tolerance's security degradation (at least in the face of credential stuffing attacks) can be largely eliminated while still preserving some functionality (e.g., for 22% of users), offering a security-focused operating point that is of theoretical interest (albeit likely impractical).

Table 10 in the Appendix provides the model performance when the training data label threshold is 25%. This label threshold prioritizes functionality, as only distinct passwords that are often susceptible (in over a quarter of the instances) are labeled as such. As a consequence, we generally witness low model susceptibility prediction rates (all below 51%) but high FNRs (as high as 85%). For example, the C_{Top2} model operating at a 50% recall level predicts susceptibility for only 8% of emails while exhibiting a 49% FNR. This model configuration significantly reduces (but does not largely eliminate) the security impact of the C_{Top2} typo-tolerance policy, with limited functionality loss.

Ultimately, this holdout test set evaluation revealed that these models offer different functionality versus security tradeoffs than with fully enabling or disabling typo-tolerance. While we conducted an initial exploration here, we believe such models are of value in practice. Note that we lack real-world user data on password typo behaviors, preventing a precise analysis of the usability impact that our machine learning models have on typo-tolerant password authentication systems. However, we can determine the lower bound impact by analyzing the user passwords that could be affected by a typo-tolerance corrector (i.e., the passwords exhibit a structure where a user could make a typo that is fixed by a corrector). Specifically, we observe that when randomly selecting a password for each user from the holdout test dataset, 86% of users have a password that could be typo-corrected by C_{Top1} and C_{Top2}[6]. For C_{Top3}—C_{Top5}, 100% of users have passwords that could be typo-corrected, as every password could potentially exhibit a typo correctable by the *rm_last* corrector (present in all three typo-tolerance policies). Thus, the vast majority of users' passwords could exhibit a correctable typo, yet our models only disable typo-tolerance for a minority of users, indicating that a significant portion of users could still benefit from typo-tolerant authentication even with our models deployed (although we leave it to future work to analyze what fraction of users precisely benefit in practice given real-world typo behaviors).

5.3.3 Computational Costs. We evaluate the computational costs of our machine learning models on a 48-core Ubuntu server with 256 GBs of memory. We find that our models can be efficiently trained and deployed for practical use by major online services.

Model Training. As discussed in Section 5.3.1, we constructed our training dataset using the BreachCompilation data, consisting of 1.4B credentials associated with 463M unique passwords (as detailed in Section 3). Processing that data to extract password features and labels for our full training dataset required approximately 12 hours total, using 20 parallel threads. Training the model for a given typo-tolerance policy took one hour on a single process. Thus, the model training process can be completed quickly even without extensive computational resources. Similarly, the model can be efficiently retrained, such as if to occasionally incorporate data from newly released breaches. (Note, incorporating new data into the training dataset would require less time than our original training dataset construction, as only the new data would need to be processed.)

[6] C_{Top2} consists of the *swc_all* and *swc_first* correctors, while C_{Top1} consists of only *swc_all*. However, any password that could exhibit a typo correctable by *swc_first* could also exhibit a typo correctable by *swc_all*. As a consequence, C_{Top1} and C_{Top2} exhibit identical proportions of users with typo-correctable passwords.

Model Deployment. The resulting decision tree model is efficient at classification. Deploying the model as a single process, it can extract a password's feature vector and produce a classification label at a rate of approximately 10K passwords per minute. Note that the model is only needed when users create or change their passwords (to determine whether typo-tolerance should be enabled for the users), and not for user logins (which occur more frequently). In addition, classification throughput can be scaled up by running multiple instances of the model in parallel. Thus, our model can be practically deployed even at the scale of major online services such as Facebook.

6 CONCLUSION

In this paper, we reconsidered the security impact of typo-tolerant password authentication. The prior analysis of such schemes [7] demonstrated that they provide notable authentication usability benefits, without affording a significant advantage to password spraying attacks. In our investigation, we considered a more comprehensive threat model that expands beyond password spraying, to additionally account for credential stuffing and tweaking attacks. Using password leak datasets, we empirically evaluated the security implications of typo-tolerant password schemes under the broader threat model. Initially, we replicated the prior findings that password spraying attacks gained little from typo-tolerance, with at most 1.24% of users becoming newly vulnerable once a typo-tolerant scheme was enabled. However, credential stuffing and tweaking attacks were significantly strengthened under typo-tolerance. Under realistic settings, up to 8.8% of users became newly vulnerable with typo-tolerance deployed, and as many as a third of users were negatively affected in the worst-case scenario. Thus, the security degradation of typo-tolerance is significantly more severe than previously understood.

To mitigate the security costs of typo-tolerant password authentication while maintaining its functionality, we explored the development of machine learning models that could predict if a password would likely become newly vulnerable to credential stuffing attacks once a typo-tolerance policy is enabled. Such models could be used to selectively disable typo-tolerance for users of predicted susceptible passwords, trading functionality for security. Our resulting models exhibit suitable operating points on the functionality-security tradeoff spectrum, offering online services the opportunity to deploy typo-tolerance in a different capacity.

Future work can expand upon our initial investigation into hardening typo-tolerant password authentication. While we considered multiple machine learning algorithms and model parameters, broader exploration of other machine learning approaches could produce models offering better functionality-security tradeoffs. Moreover, future work can investigate the exact usability impact of such models, by considering the extent to which these models preserve typo-tolerance for users who actually make password typos. Additionally, our models focused on limiting the impact of credential stuffing attacks on typo-tolerant schemes, and further work is needed to address credential tweaking attacks. More recently, a personalized variant of typo-tolerance was developed [8] that learns to correct the specific password typos of individual users over time, providing further usability benefits compared to vanilla typo-tolerance, without significant security degradation against online and offline brute-force guessing attacks. Revisiting the security analysis of this variant under a similar threat model as used in this paper would be interesting, as credential stuffing and tweaking attacks may similarly benefit from the typo corrections, but to what extent is unclear given the personalized nature of the corrections. Finally, researchers could explore whether online services using typo-tolerant password authentication may be able to better protect their users by disallowing password changes/resets where the old password could be corrected to the new one under the typo-tolerance policy used. Such an approach could help prevent old or leaked passwords from being successfully leveraged against a typo-tolerant system during credential stuffing attacks.

7 ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation award CNS-2055549, for which we are grateful. The opinions expressed in this paper do not necessarily reflect those of the research sponsors.

REFERENCES

[1] 4iQ. 2020. Weaponized Data Breaches: Fueling Identity-based Attacks Across the Globe. https://https://4iq.com/2020-identity-breach-report/.
[2] Susan Antilla. 2015. Is Vanguard Making It Too Easy for Cybercriminals to Access Your Account? https://www.thestreet.com/opinion/is-vanguard-making-it-too-easy-for-cybercriminals-to-access-your-account-13213265.
[3] Bijeeta Pal. 2019. Password Similarity Models using Neural Networks. https://github.com/Bijeeta/credtweak/tree/master/credTweakAttack.
[4] Enka Blanchard. 2020. Making More Extensive and Efficient Typo-Tolerant Password Checkers. In *IEEE Annual Computers, Software, and Applications Conference (COMPSAC)*.
[5] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symposium on Security and Privacy (S&P)*.
[6] Julio Casal. 2017. 1.4 billion cleartext credentials discovered in a single database. https://medium.com/4iqdelvedeep/1-4-billion-clear-text-credentials-discovered-in-a-single-database-3131d0a1ae14.
[7] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. 2016. pASSWORD tYPOS and How to Correct Them Securely. In *IEEE Symposium on Security and Privacy (S&P)*.
[8] Rahul Chatterjee, Joanne Woodage, Yuval Pnueli, Anusha Chowdhury, and Thomas Ristenpart. 2017. The TypTop System: Personalized Typo-Tolerant Password Checking. In *ACM Conference on Computer and Communications Security (CCS)*.
[9] Katie Collins. 2017. Facebook buys black market passwords to keep your account safe. https://www.cnet.com/news/facebook-chief-security-officer-alex-stamos-web-summit-lisbon-hackers/.
[10] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The Tangled Web of Password Reuse. In *Network and Distributed System Security Symposium (NDSS)*.
[11] Xavier de Carne de Carnavalet and Mohammad Mannan. 2014. From Very Weak to Very Strong: Analyzing Password-Strength Meters. In *Network and Distributed System Security Symposium (NDSS)*.
[12] Dropbox. 2016. zxcvbn. https://github.com/dropbox/zxcvbn.
[13] Maximilian Golla, Miranda Wei, Juliette Hainline, Lydia Filipe, Markus Dürmuth, Elissa Redmiles, and Blase Ur. 2018. "What was that site doing with my Facebook password?": Designing Password-Reuse Notifications. In *ACM Conference on Computer and Communications Security (CCS)*.
[14] Hana Habib, Jessica Colnago, William Melicher, Blase Ur, Sean Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Cranor. 2017. Password Creation in the Presence of Blacklists. In *Workshop on Usable Security and Privacy (USEC)*.
[15] Josh Hendrickson. 2019. Facebook Fudges Your Password for Your Convenience. https://www.howtogeek.com/402761/facebook-fudges-your-password-for-your-convenience/.
[16] Cormac Herley and Stuart E Schechter. 2019. Distinguishing Attacks from Legitimate Authentication Traffic at Scale. In *Network and Distributed System Security Symposium (NDSS)*.
[17] Troy Hunt. 2020. The 773 Million Record "Collection #1" Data Breach. https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/.

[18] Mark Keith, Benjamin Shao, and Paul Steinbart. 2009. A Behavioral Analysis of Passphrase Design and Effectiveness. *Journal of the Association for Information Systems* 10 (02 2009), 63–89.

[19] Mark Keith, Benjamin Shao, and Paul John Steinbart. 2007. The usability of passphrases for authentication: An empirical field study. *International Journal of Human-Computer Studies* 65, 1 (2007), 17 – 28.

[20] Lucy Li, Bijeeta Pal, Junade Ali, Nick Sullivan, Rahul Chatterjee, and Thomas Ristenpart. 2019. Protocols for Checking Compromised Credentials. In *ACM Conference on Computer and Communications Security (CCS)*.

[21] MITRE. 2020. Credential Stuffing. https://attack.mitre.org/techniques/T1110/004/.

[22] MITRE. 2020. Password Spraying. https://attack.mitre.org/techniques/T1110/003/.

[23] Moni Naor, Benny Pinkas, and Eyal Ronen. 2019. How to (not) Share a Password: Privacy Preserving Protocols for Finding Heavy Hitters with Adversarial Behavior. In *ACM Conference on Computer and Communications Security (CCS)*.

[24] OWASP. 2020. Credential Stuffing. https://owasp.org/www-community/attacks/Credential_stuffing.

[25] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart. 2019. Beyond Credential Stuffing: Password Similarity Models using Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*.

[26] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. 2017. Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In *ACM Conference on Computer and Communications Security (CCS)*.

[27] Shannon Riley. 2006. Password security: What users know and what they actually do. *Usability News* 8, 1 (2006), 2833–2836.

[28] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. 2010. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *USENIX Conference on Hot Topics in Security (HotSec)*.

[29] Stuart Schechter, Yuan Tian, and Cormac Herley. 2019. StopGuessing: Using Guessed Passwords to Thwart Online Guessing. In *IEEE European Symposium on Security and Privacy (EuroS&P)*.

[30] Shape. 2020. Credential Spill Report. https://federalnewsnetwork.com/wp-content/uploads/2020/06/Shape-Threat-Research-Credential-Spill-Report.pdf.

[31] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2010. Encountering stronger password requirements: user attitudes and behaviors. In *USENIX Symposium on Usable Privacy and Security (SOUPS)*.

[32] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, Daniel Margolis, Vern Paxson, and Elie Bursztein. 2017. Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. In *ACM Conference on Computer and Communications Security (CCS)*.

[33] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, et al. 2019. Protecting Accounts from Credential Stuffing with Password Breach Alerting. In *USENIX Security Symposium*.

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

[35] Chun Wang, Steve TK Jan, Hang Hu, Douglas Bossart, and Gang Wang. 2018. The next domino to fall: Empirical analysis of user passwords across online services. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*.

[36] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. 2016. Targeted Online Password Guessing: An Underestimated Threat. In *ACM Conference on Computer and Communications Security (CCS)*.

[37] Ke Coby Wang and Michael K Reiter. 2020. Detecting Stuffing of a User's Credentials at Her Own Accounts. In *USENIX Security Symposium*.

[38] Yinqian Zhang, Fabian Monrose, and Michael K Reiter. 2010. The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis. In *ACM Conference on Computer and Communications Security (CCS)*.

Password Leak Datasets	Password Reuse Rate (%)	No. of Total Distinct Emails	No. of Distinct Emails w/ Multiple Passwords
*BreachCompilation	20.9	1.1B	182.1M
Collection#1: BT_Combos	87.0	18.9M	3.2M
Collection#1: Dumps_dehashed	98.6	4.5M	1.8M
*Collection#1: EUcombos	49.8	186.6M	23.5M
*Collection#1: EUcombos_1	30.7	125M	13.3M
*Collection#1: Gamescombos	30.1	67.5M	6.9M
Collection#1: Gamescombos_Dumps	53.6	92.4M	9M
Collection#1: Gamescombos_Sharpening	84.4	120.1M	27.4M
Collection#1: MAILACCESScombos	81.9	26M	7.5M
Collection#1: Monetarycombos	64.6	4.9M	1.5M
Collection#1: NEWcombosemiprivate_Dumps	89.2	7.5M	1.9M
*Collection#1: NEWcombosemiprivate_EUcombo	< 0.1	313.1M	27.8M
Collection#1: NEWcombosemiprivate_Privatecombos	95.1	301.7M	171.3M
Collection#1: NEWcombosemiprivate_UpdateDumps	72.7	23.8M	1.2M
Collection#1: Numberpasscombos	50.0	3285	15
*Collection#1: OLDCLOUD_BTCcombos	47.6	6.6M	1.5M
Collection#1: OLDCLOUD_CHINAcombos	62.9	13.6M	1.7M
Collection#1: OLDCLOUD_Dumpcleaned-deletedduplicate	85.0	8.7M	2.9M
Collection#1: OLDCLOUD_Gamingcombos	63.5	84.2M	9.4M
Collection#1: OLDCLOUD_Hackingcombo	27.5	449K	4266
Collection#1: OLDCLOUD_Japancombos	96.4	12.5M	7M
Collection#1: OLDCLOUD_Monetarycombos	58.5	20.1M	3.7M
Collection#1: OLDCLOUD_OLDDUMPSDEHASHED	57.8	109.3M	15.2M
*Collection#1: OLDCLOUD_Porncombos	11.4	4.5M	257K
Collection#1: OLDCLOUD_Shoppingcombos	84.0	12.8M	2.1M
Collection#1: OLDCLOUD_Tradingcombos	0.7	455K	6826
*Collection#1: OLDCLOUD_UKcombos	11.8	15.9M	2M
Collection#1: OLDCLOUD_USAcombos	90.0	28M	7M
*Collection#1: RUcombo	7.5	18.1M	3.7M
Collection#1: Shoppingcombos	52.5	4.4M	240K
Collection#1: USAcombos	81.9	26M	7.5M
Collection#1: USERPASScombos	25.0	54.7K	12

Table 6: For all password leak datasets that we collected, we list the password reuse rates (i.e., the percent of emails associated with the same password multiple times, out of all emails associated with multiple passwords), as well as the number of total unique emails and the number of distinct emails associated with multiple passwords. As discussed in Section 3, we do not investigate leaks with few (<10K) multi-password emails or those exhibiting a password reuse rate exceeding 50% (as such reuse rates are not commensurate with prior findings [10, 32], indicating likely data duplication). We indicate which leaks we use for our study with an asterisk (*).

Dataset	Policy	q = 10				q = 100				q = 1000			
		All		Multi		All		Multi		All		Multi	
		Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
BreachCompilation	C_{None}	1.73	1.34	2.62	0.22	3.24	2.54	4.87	0.61	6.73	5.25	10.48	1.52
	C_{Top5}	+0.10	+0.07	+0.20	+0.02	+0.14	+0.12	+0.23	+0.06	+0.38	+0.31	+0.66	+0.22
C#1: EUcombos	C_{None}	2.09	1.87	2.52	0.73	3.93	3.54	4.53	1.41	8.39	7.51	10.35	3.32
	C_{Top5}	+0.03	+0.03	+0.05	+0.03	+0.18	+0.17	+0.25	+0.17	+0.48	+0.47	+0.70	+0.66
C#1: EUcombos_1	C_{None}	2.29	2.05	2.80	0.50	4.11	3.68	5.16	1.12	8.96	8.04	11.15	2.48
	C_{Top5}	+0.04	+0.03	+0.05	+0.02	+0.19	+0.18	+0.29	+0.22	+0.49	+0.47	+0.83	+0.66
C#1: Gamescombos	C_{None}	2.24	1.99	3.05	0.57	4.20	3.71	5.97	1.13	8.07	7.12	11.35	2.04
	C_{Top5}	+0.18	+0.16	+0.21	+0.05	+0.18	+0.16	+0.25	+0.10	+0.41	+0.38	+0.62	+0.33
C#1: NEW_csp_EUcombo	C_{None}	2.37	2.01	4.10	0.08	4.09	3.50	6.95	0.28	7.82	6.71	13.21	0.78
	C_{Top5}	+0.02	+0.01	+0.05	+0.01	+0.15	+0.13	+0.30	+0.06	+0.33	+0.29	+0.64	+0.19
C#1: OC_BTCcombos	C_{None}	1.20	0.78	2.77	0.92	2.57	1.72	5.78	2.01	5.66	3.90	11.98	4.26
	C_{Top5}	+0.11	+0.08	+0.26	+0.15	+0.11	+0.09	+0.21	+0.12	+0.35	+0.28	+0.58	+0.27
C#1: OC_Porncombos	C_{None}	2.75	2.44	6.08	0.67	5.22	4.69	10.48	1.29	11.68	10.70	19.85	2.89
	C_{Top5}	+0.05	+0.04	+0.11	+0.04	+0.22	+0.21	+0.47	+0.18	+0.48	+0.45	+0.94	+0.44
C#1: OC_UKcombos	C_{None}	1.64	1.34	2.41	0.07	4.39	3.58	7.00	0.56	9.95	8.09	16.72	2.06
	C_{Top5}	+0.13	+0.11	+0.18	+0.02	+0.23	+0.20	+0.41	+0.15	+0.72	+0.63	+1.24	+0.58
C#1: RUcombo	C_{None}	6.40	1.96	23.81	1.44	9.61	3.66	31.60	1.64	13.55	6.69	36.75	2.20
	C_{Top5}	+0.37	+0.23	+0.72	+0.02	+0.33	+0.21	+0.64	+0.04	+0.49	+0.33	+0.88	+0.09

Table 7: The effectiveness of password spraying attacks when password typo-tolerance is disabled (C_{None}) compared with using the C_{Top5} typo-tolerance policy, across different datasets and different numbers of attack queries q. For each leak, we evaluate attack success on all emails (labeled as *All*) and only emails with multiple passwords (labeled as *Multi*), using the upper bound and lower bound attack success metrics. For ease of comparison, the C_{Top5} attack success metrics are the percentage point increases/deltas (as indicated by the + sign) in password spraying success over C_{None}, rather than the total attack success rate.

Feature Type	Index	Description
Categorical Features	1-15	The combination of character classes appearing in the password, considering four classes (uppercase letters, lowercase letters, digits, and symbols)
	16-19	Character class of the first password character
	20-23	Character class of the last password character
Numerical Features	24	Password length
	25	Password Shannon entropy
	26	Password strength (as outputted by the zxcvbn password strength meter [12])
	27	Password popularity/rank
	28	Number of character classes appearing in the password
	29	Number of character class transitions in the password
	30	Number of character class transitions divided by password length
	31	Proportion of the password for the longest single-character-class substring
	32	Proportion of the password for the shortest single-character-class substring
	33	Number of characters of the same class at the end of the password
	34	Longest sequence of repeated characters in the password
	35	Proportion of the password for the longest sequence of repeated characters
Boolean Features	36	Last character is uppercase letter or symbol
	37	Last two characters are identical
	38	Last two characters are sequential digits or letters (e.g., 12 or "ab")
	39	Shift-key modified last character is identical to the second-to-last character
	40	Second-to-last character and the shift-key modified last character are sequential
	41	First two characters are identical
	42	First two characters are sequential
	43	Case-swapped first character is identical to the second character
	44	Case-swapped first character and the second character are sequential
	45	First two characters are the same character class
	46	Last two characters are the same character class

Table 8: The description of password features used for our machine learning models.

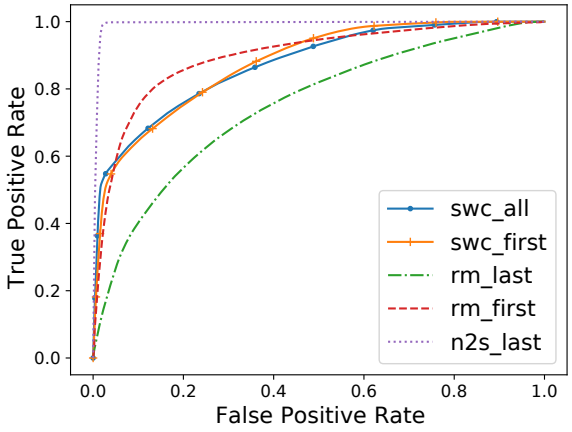

Figure 3: ROC curves for the individual corrector models trained on data labeled using a 10% label threshold.

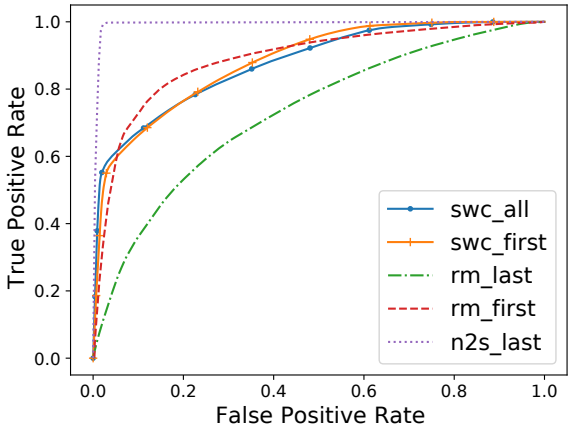

Figure 4: ROC curves for the individual corrector models trained on data labeled using a 25% label threshold.

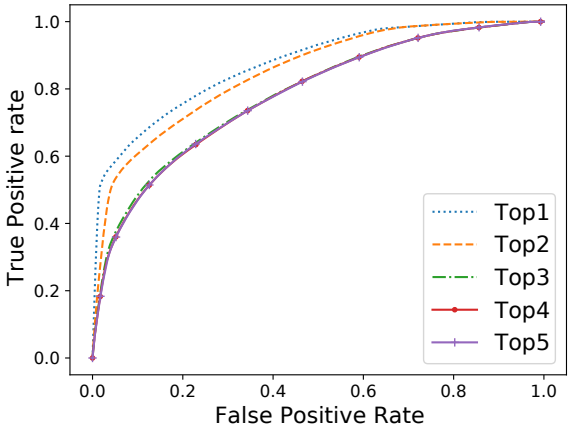

Figure 5: ROC curves for the policy models trained on data labeled using a 10% label threshold.

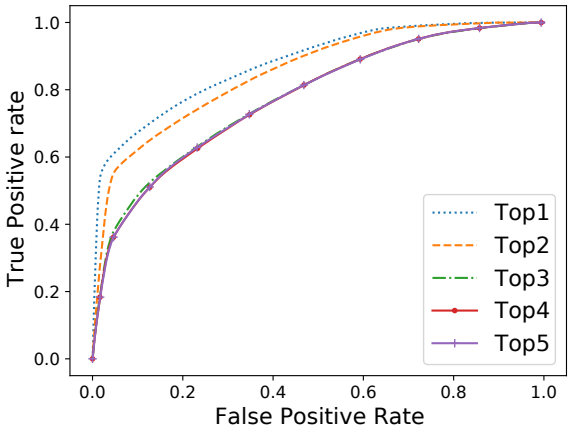

Figure 6: ROC curves for the policy models trained on data labeled using a 25% label threshold.

Label Threshold	Policy	Recall	Susceptible	FPR	FNR
0%	Top1	10%	0.63	0.63	0.35
		25%	0.68	0.68	0.27
		50%	0.72	0.72	0.15
		75%	0.75	0.75	0.11
		90%	0.78	0.78	0.09
	Top2	10%	0.70	0.71	0.38
		25%	0.72	0.72	0.17
		50%	0.73	0.73	0.08
		75%	0.75	0.75	0.05
		90%	0.78	0.78	0.03
	Top3	10%	0.79	0.78	0.15
		25%	0.79	0.79	0.10
		50%	0.80	0.80	0.08
		75%	0.88	0.88	0.04
		90%	0.93	0.93	0.02
	Top4	10%	0.78	0.78	0.15
		25%	0.79	0.78	0.11
		50%	0.84	0.83	0.07
		75%	0.89	0.88	0.04
		90%	0.94	0.94	0.01
	Top5	10%	0.63	0.63	0.24
		25%	0.64	0.63	0.18
		50%	0.78	0.78	0.9
		75%	0.87	0.87	0.04
		90%	0.94	0.93	0.01

Table 9: The performance of the password classifier models on the holdout test dataset, trained on data labeled using a 0% label threshold. We evaluate the five policy models each tuned to varying recall operating points, and determine the proportion of emails whose randomly selected password is flagged as susceptible by our models, as well as the models' false positive and false negative rates.

Label Threshold	Policy	Recall	Susceptible	FPR	FNR
25%	Top1	10%	0.03	0.03	0.70
		25%	0.04	0.03	0.64
		50%	0.04	0.04	0.53
		75%	0.31	0.31	0.23
		90%	0.51	0.51	0.14
	Top2	10%	0.06	0.05	0.71
		25%	0.07	0.06	0.60
		50%	0.08	0.07	0.49
		75%	0.27	0.26	0.35
		90%	0.37	0.36	0.29
	Top3	10%	0.05	0.04	0.81
		25%	0.05	0.04	0.76
		50%	0.13	0.11	0.64
		75%	0.28	0.26	0.54
		90%	0.35	0.34	0.50
	Top4	10%	0.05	0.04	0.81
		25%	0.06	0.05	0.76
		50%	0.13	0.11	0.65
		75%	0.29	0.28	0.54
		90%	0.43	0.43	0.49
	Top5	10%	0.03	0.02	0.85
		25%	0.04	0.03	0.80
		50%	0.11	0.10	0.66
		75%	0.25	0.24	0.57
		90%	0.40	0.40	0.50

Table 10: The performance of the password classifier models on the holdout test dataset, trained on data labeled using a 25% label threshold. We evaluate the five policy models each tuned to varying recall operating points, and determine the proportion of emails whose randomly selected password is flagged as susceptible by our models, as well as the models' false positive and false negative rates.