# VersaPSE: Versatile Password Strength Evaluation Using Continual Learning

Yifei Zhang[1], Zhenduo Hou[2,(✉)], and Ding Wang[2]

[1] College of Computer Science, Nankai University
[2] College of Cryptology and Cyber Science, Nankai University
joehou13@nankai.edu.cn

**Abstract.** Credential tweaking attacks leverage a user's leaked passwords and create their variants to attempt logins to the user's other accounts, achieving a success rate of over 70% with only $10^3$ online guesses. Reuse-based password strength meters (reuse-based PSMs), often derived from credential tweaking models, are proposed to measure password strength against such attacks. However, the performance of these reuse-based PSMs has *never been systematically evaluated and compared*, leaving their effectiveness in real-world attack scenarios in question.

In this work, we conduct systematic evaluations of existing reuse-based PSMs and find that they are highly inaccurate and often incur high latency. To address this issue, we propose VersaPSE, the first versatile reuse-based password strength evaluation framework for online guessing attacks. Under VersaPSE, credential tweaking models can be adapted into *multipurpose* PSMs via continual learning techniques, enabling accurate strength evaluation under both credential tweaking and trawling guessing attacks. Experiments on 12 large-scale datasets show that under VersaPSE, our five multipurpose PSMs derived from credential tweaking models achieve 3.8%∼36.7% higher accuracy than leading reuse-based PSMs against credential tweaking attacks, and outperform existing PSMs under online trawling guessing attacks. We also provide a proof-of-concept implementation of our VersaPSE. This work not only introduces a novel technical route for designing effective PSMs tailored to credential tweaking attacks, but also extends the use of credential tweaking models in defending against online trawling guessing attacks. Our artifact is available at https://github.com/FeliceRivarez/VersaPSE_code.

**Keywords:** Password strength meters, Password behaviors, Credential tweaking attacks, Continual Learning

## 1 Introduction

Passwords remain the mainstream form of user authentication due to their low deployment costs, ease of change, and remarkable simplicity [17, 27, 83]. Due to limited memorability, users are prone to reusing passwords and using weak passwords [5,10,38], making passwords vulnerable to credential tweaking attacks (i.e., modifying a user's leaked password in guessing a new one [51,70,71,79,80]) and trawling (requiring no personal information [38,72,84]) guessing attacks. To

resist such online attacks, almost every respectable service deploys a password strength meter (PSM) to nudge users towards stronger passwords [31, 68].

Despite their importance, existing PSMs fall short of accurately evaluating password strengths against various online guessing attacks. To begin with, deployed PSMs (mostly industry-proposed, e.g., Zxcvbn-PSM [77] and 12306-PSM [8]) mainly focus on evaluating strengths against trawling guessing attacks, leaving credential stuffing and tweaking attacks largely untouched. In these attacks, an attacker can directly use and slightly modify a user's leaked password from one service (i.e., a sister password) to more efficiently guess (e.g., logging in without the user's permission) a user's password at another service (e.g., `alice123`→`Alice123!`). Since 62%∼85% of users reuse passwords [6, 12, 35] and unending password breaches (e.g., [21, 29, 62, 78]) make sister passwords more accessible, credential stuffing and tweaking attacks become increasingly threatening. They not only account for over 34% of online login attempts [44, 57], but can also achieve a success rate of 70% with $10^3$ online guesses, causing severe consequences. For instance, in 2022, the credential stuffing attack against the betting service DraftKings resulted in $30,000 in financial losses from about 67,000 victims [55]. *Therefore, it is pressing to design accurate reuse-based PSMs to evaluate password strengths against credential tweaking attacks.*

Nevertheless, leading reuse-based PSMs (e.g., Vec-PPSM [51] and KNN-PSM [42]) remain unsatisfactory. Broadly, reuse-based PSMs can be categorized into two types. The first similarity-based approach uses fixed similarity metrics (e.g., FastText [15, 46]) to detect password reuse, which can be susceptible to high false positive rates. For example, `fantasy521478` can resist (i.e., without being cracked within $10^3$ guesses) state-of-the-art credential tweaking models [51, 70, 71, 79, 80] even if `521478` is leaked, yet similarity-based Vec-PPSM [51] deems it insecure. Such a similarity-based approach falls short of capturing complex users' real-world edit operations (e.g., deletions and insertions), and exhibits inconsistencies across various similarity metrics (e.g., Embedding similarity, edit distance, and cosine similarity) [19, 34, 71, 79]. The second enumeration-based approach deems a number of (e.g., $10^3$) sister password variants generated by the corresponding credential tweaking model insecure, and can be susceptible to high false negative rates. This is mainly because passwords deemed secure by one model might be cracked by another. For instance, BERT-PSM [80] evaluates `19caesar` as secure when `caesar` is leaked, yet TarGuess-II [70] can crack it within $10^3$ guesses. Beyond these flaws, there exist no systematic evaluations of major reuse-based PSMs, and their deployability (e.g., computation and storage overheads) remains unknown, casting doubt on their real-world effectiveness.

### 1.1 Motivations and challenges

Firstly, the accuracy of existing reuse-based PSMs has not been systematically evaluated. For example, in terms of accuracy, Vec-PPSM [51], which is the first reuse-based PSM, has been evaluated only on a single password dataset; PR-PSM [79] was only compared with a trawling PSM (Zxcvbn-PSM [77]), and it has not been compared with other reuse-based PSMs, leaving the important accuracy issue understudied. Worse still, BERT-PSM [80] and KNN-PSM [42] have

never been evaluated in terms of either accuracy or deployability (e.g., computation and storage overhead). This calls for a principled approach to evaluate the accuracy of reuse-based PSMs, as only accurate PSMs can strengthen password security, while inaccurate ones do more harm than good [28, 31, 68].

Secondly, existing approaches for deriving reuse-based PSMs from credential tweaking models rely heavily on heuristics and are computationally intensive. Essentially, existing PSMs mainly utilize credential tweaking models and deem a user's password insecure/vulnerable only if it is cracked [42, 79, 80]. This results in overestimating password strength, as only a very limited fraction of unsafe passwords can be cracked by a single credential model [71, 79]. Passwords deemed secure by such PSMs can still be vulnerable, as attackers can use more advanced credential tweaking models than those employed by the PSMs. Besides, conducting a credential tweaking attack is computationally intensive [42], limiting the deployability of PSMs that conduct such an attack when evaluating password strength. This calls for a principled approach to derive accurate and computationally efficient reuse-based PSMs from credential tweaking models.

Thirdly, existing studies show that capturing password reuse behavior is particularly beneficial for measuring password strength against not only credential tweaking attacks, but also trawling guessing attacks. This finding is confirmed by the success of fuzzyPSM [67], one of the most prominent trawling PSMs to date, as it captures password reuse behavior between user groups to evaluate password strength under trawling guessing attacks. However, though credential tweaking models are powerful models that capture the complex behavior of password reuse, they are only utilized to defend users against credential tweaking attacks. How to extend the capability of credential tweaking models to evaluate password strength under trawling guessing attacks is an interesting topic.

### 1.2 Contributions

- **New findings on users' vulnerable password behaviors.** Through observation of real-world password datasets, we find that reusing (or slightly modifying passwords) across different services and using popular passwords can be viewed as the vulnerable behaviors of reusing locally and globally popular passwords, respectively, and they exhibit different levels of complexity. This enables the application of credential tweaking models to password strength evaluations against both credential tweaking and trawling guessing attacks under a unified continual learning framework, with each task focusing on a different reuse behavior corresponding to the respective guessing attack.
- **A new technical route for multipurpose password strength meters.** Building on our key findings, we propose VersaPSE, a versatile password strength evaluation (PSE) framework for designing multipurpose password strength meters (PSMs) using credential tweaking models. By employing continual learning techniques, VersaPSE treats evaluating password strengths against more complex credential tweaking attacks as the initial task, and trawling guessing attacks as the subsequent task. This enables a single credential tweaking model to capture the impacts of both local and global reuses, and consequently, accurate password strength evaluations.

– **Extensive evaluations.** Using VersaPSE, we design five multipurpose PSMs from the corresponding mainstream credential tweaking models (i.e., PointerGuess [79], Pass2Edit [71], PassBERT [80], Pass2Path [51], and KNNGuess [42]). We then conduct extensive evaluations that involve 12 large-scale password datasets and 10 mainstream password strength meters, including both reuse-based PSMs (e.g., PR-PSM [79], BERT-PSM [80], and KNN-PSM [42]) and trawling-based PSMs (e.g., Zxcvbn-PSM [77], KeePSM [56], Microsoft-PSM [9] and 12306-PSM [8]). Experiment results demonstrate the superior performance of our five multipurpose PSMs derived under the VersaPSE framework, as they all outperform existing reuse-based and trawling PSMs in both credential tweaking and trawling guessing attack scenarios.
– **Some discussions.** We show that VersaPSE can be deployed in a lightweight approach at the client side, with sub-second evaluation (per-password) and modest storage on an off-the-shelf computer. We also provide a proof-of-concept implementation of our VersaPSE as a browser extension, demonstrating the practical deployability. Moreover, we further discuss practical applications (e.g., in password managers) and limitations of our VersaPSE.

## 2 Preliminaries and related work

We mainly focus on evaluating password strengths against credential tweaking and trawling guessing attacks. For each type, we present attack models and their corresponding password strength meters (PSMs). We do not consider offline guessing attacks because they assume attackers have almost unlimited guesses against users' passwords, which are fundamentally different from online guessing.

### 2.1 Credential tweaking attacks

Credential stuffing and tweaking attacks leverage a user's leaked password $pw^s$ (i.e., a sister password) on one service (e.g., Twitter) to more efficiently guess the targeted password $pw$ on another one (e.g., LinkedIn) [42,51,52,70,71,79]. Since credential stuffing attacks using an exact sister password can be mitigated by leakage detection services like HIBP [11], we mainly focus on credential tweaking attacks that use variants of sister passwords (e.g., `alice123`→`Alice123!`).

In 2016, Wang et al. proposed the first credential tweaking model, TarGuess-II [70], which parses passwords into letter, digit, and symbol structures and allows deletions and insertions at both structure and character levels. In 2019, Pal et al. proposed the first *deep-learning-based* credential tweaking model Pass2Path [51], which utilizes a sequence-to-sequence framework [60] to more adaptively learn and predict character-level edit operations (i.e., insertions and deletions) in modifying passwords. In 2023, Wang et al. [71] proposed the RNN-based Pass2Edit model that treats credential tweaking attacks as a series of multi-class classification tasks of users' edit operations. Similar to [70], it incorporates popular passwords, and outputs a hybrid guessing dictionary containing popular passwords. Concurrently, Xu et al. proposed PassBERT [80], a bi-directional Transformer [24] that follows a pre-training and fine-tuning paradigm, to guess passwords in different attack scenarios: PassBERT first trains a general password-guessing architecture and then fine-tunes models for specific (including credential

tweaking) attacks [80]. In 2024, Xiu and Wang proposed PointerGuess [79], which uses the pointer mechanism in natural language processing [58, 65] to simulate the behavior of copying segments from sister passwords (e.g, copying `MacOP` from `IloveMacOP` in generating `MacOP6789`). In 2026, Li and Wang further proposed KNNGuess [42], which uses $k$-nearest-neighbor retrieval to simulate users' behavior of replacing password components with semantically related ones with preserved structure (e.g., `Shark0301`→`Bear03`).

To resist such attacks, various reuse-based PSMs, which are mainly developed from credential tweaking models, have been proposed. These PSMs are broadly categorized into similarity- and enumeration-based types. In 2019, Pal et al. [51] introduced the similarity-based Vec-PPSM that uses the FastText-based embedding similarity [15] in evaluating password strengths, and a high similarity between the examined and sister passwords indicates low password strengths (i.e., insecurity) against the Pass2Path attack model. However, Vec-PPSM only considers character-level similarity (e.g., `alice123` is similar to `123456`), without complex real-world edit operations that have been incorporated into the Pass2Path model, so its accuracy is dubious (see more details in Sec. 5.3). Enumeration-based PSMs, such as KNN-PSM, PR-PSM, and BERT-PSM, employ their corresponding credential tweaking models like KNNGuess [42], PointerGuess [79], and PassBERT [80] to generate a number of (e.g., $10^3$) variants of sister passwords: A password is deemed vulnerable/insecure if it belongs to the generated variants. However, such PSMs can overestimate password strength because one credential tweaking model covers only a small proportion of vulnerable passwords, whereas powerful (yet realistic) attackers may utilize a different and more advanced model to cover a wider range of variants.

### 2.2 Online trawling guessing attacks

A trawling guessing attack can operate without a user's sister password. Due to users' vulnerable behavior of using popular passwords (e.g., `abc123`) and attackers' capabilities of employing advanced guessing models (e.g., PCFG [76], RFGuess [72], RankGuess [81], and PassLLM [84]), online trawling guessing attacks targeting popular passwords are also of practical concern.

To nudge users towards more secure passwords against trawling guessing attacks, almost all respectable services deploy trawling-based PSMs. On the one hand, industry-proposed PSMs mainly employ rule- and pattern-based PSMs due to their simplicity. A rule-based PSM (e.g., Microsoft-PSM [9] and 12306-PSM [8]) checks whether a password satisfies given composition rules such as minimum lengths and character types, while a pattern-based PSM (e.g., Zxcvbn-PSM [77] and KeePSM [56]) checks whether a password contains common patterns like keyboard strikes and repeated characters and then estimates its guess numbers. For instance, Zxcvbn-PSM identifies the password `JohnSmith123` as the combination of the common username `JohnSmith` and digit sequence `123`, calculates the minimum guess number needed to crack each pattern, and combines them to estimate the total guess number. Zxcvbn-PSM is widely deployed by services such as Dropbox [3], Bitwarden [1], and WordPress [2], and has been recognized as one of the most promising industry-proposed PSMs [31, 67, 68].

**Table 1:** Password datasets considered in this work

| Dataset | Language | Service | Total PWs | Leaked time | Email | Invalid | Cleaned% |
|---|---|---|---|---|---|---|---|
| Dodonew | Chinese | E-commerce | 16,282,286 | Dec. 2011 | Yes | 256,016 | 1.57% |
| Tianya | | Social Forum | 30,816,592 | Dec. 2011 | No | 9,062 | 0.03% |
| CSDN | | Program forum | 6,428,410 | Dec. 2011 | Yes | 3,164 | 0.05% |
| Taobao | | E-commerce | 15,072,418 | Feb. 2016 | Yes | 1,265 | 0.01% |
| 126 | | Email | 88,136,038 | Dec. 2011 | Yes | 14,995 | 0.23% |
| Weibo | | Social Media | 4,730,662 | Dec. 2011 | No | 458 | <0.01% |
| 000webhost | English | Web hosting | 15,299,907 | Oct. 2015 | Yes | 116,462 | 0.76% |
| LinkedIn | | Job hunting | 54,656,615 | Jan. 2012 | Yes | 122,051 | 0.22% |
| Twitter | | Social media | 25,575,929 | May. 2016 | Yes | 287,551 | 1.12% |
| MathWay | | Education | 16,051,087 | Jan. 2020 | Yes | 209,726 | 1.31% |
| Phpbb | | Tech forum | 255,421 | Jan. 2009 | No | 10 | <0.01% |
| Rockyou | | Social media | 32,575,500 | Jul. 2012 | No | 9,864 | <0.01% |

On the other hand, academia-proposed PSMs also fall into various categories. A statistical PSM (e.g., PCFG-PSM [37] and fuzzyPSM [67]) utilizes the corresponding statistical password guessing model (e.g., PCFG [76]) to output a probability as the password strength/guessability. Similarity-based PSMs such as LPSE [34] compare a password with a predetermined strong password (e.g., calculating cosine similarity between `password` and `4b#8-C+0NDIXpC[o4`), so this PSM requires no training set [34]. With the development of deep learning techniques, RNN-PSM [45] and CNN-PSM [53] have also been proposed to better capture password strengths via the generalization abilities of neural networks.

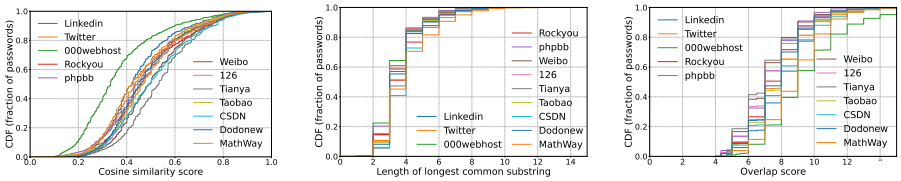
## 3 Key Findings

### 3.1 Datasets and ethical considerations

In this study, we use 12 large-scale datasets, a total of 300 million passwords (see Table 1). These datasets were leaked from 2009 to 2020 and cover a variety of services (e.g., E-commerce, social, and forums) in both English and Chinese, thus reflecting real-world user behaviors. For instance, 000webhost is mainly used by service administrators who are likely to have higher security awareness. Besides, although these datasets were leaked 6∼17 years ago, they can still reasonably capture characteristics of current passwords. This is because password distributions change only marginally over time [16], and statistical evidence also confirms that progress toward stronger passwords is limited [14].

**Data cleaning**. Since a sister password is a user's password at another service, we construct them by matching email addresses (see Table 2) across services. We remove invalid email addresses (e.g., those without '`@`'), as they cannot be used to match email addresses. Further, as in prior studies [51,71,79,80], we only consider passwords composed of 95 printable ASCII characters, and remove those with length≥30 as in prior studies [42, 71, 79, 80], since they are more likely to be created automatically by machines rather than manually by humans.

**Ethical consideration**. Although passwords in Table 1 are publicly available on the Internet and have been widely used in prior studies such as [51,71,79,80], passwords are essentially sensitive, and using them can bring potential risks to affected users. We acknowledge such risks and recognize that affected users have not consented to the use of their data in research, nor have they had any chance




2-gram cosine distribution LCS distribution Overlap distribution

**Fig. 1:** Similarities between users' passwords and *popular passwords*. For cosine similarity, LCS, and overlap score, the higher the score, the higher the similarity, and a large proportion of users' passwords are highly similar to popular ones.

to consent. In particular, we take four precautions to ensure that this study does not pose additional risks to users: (1) We do not further disseminate breached datasets to other parties and only use them for this study; (2) Our datasets are stored and processed on computers unconnected to the Internet; (3) We treat each account as confidential, and mainly report aggregated statistics such as frequencies to minimize the impacts on affected users; (4) We delete all sensitive information after our analysis. Our work meets the key ethical guidelines [61], and has obtained the IRB approval from our institution.

### 3.2 Revisiting password reuse behaviors

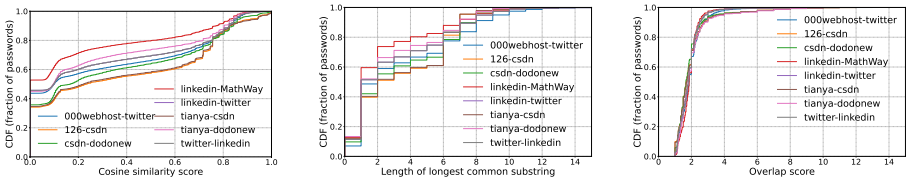
In essence, VersaPSE is built upon the following two key empirical observations of users' vulnerable password behaviors, and we present them below.

**Finding 1** *Besides reusing her locally popular sister password, a user also reuses globally popular passwords that are adopted by other users.*

*This finding provides the empirical foundation for unifying users' different vulnerable behaviors in a unified approach (e.g., via conditional probability).* More concretely, using a sister password can be seen as reusing a user-specific locally popular password, while using weak (e.g., top-$10^3$) passwords can be seen as reusing globally popular passwords. To validate global reuses, we randomly select $10^4$ passwords and compute their similarity scores with *the most likely password* among the top-$10^3$ popular ones. Without loss of generality, we employ widely used 2-gram cosine, Longest Common Substring (LCS), and overlap score (higher values mean more similarity) [19, 51, 70, 71, 79]. As shown in Fig. 1, we find that up to 74.9% of users' passwords are highly similar to the top-$10^3$ weak passwords for datasets in Table 1 when similarity thresholds are set to be 0.5 for cosine similarity, 4 for LCS, and 6 for overlap score, respectively. We also explore other similarity metrics such as Levenshtein and Manhattan distances. More detailed results are shown in Appendix C and are consistent with the above results.

**Finding 2** *The behavior of reusing locally popular sister passwords is generally more complex than reusing globally popular passwords.*

This finding reveals that local and global password reuse behaviors are inherently different: Compared to reusing globally popular passwords, reusing locally popular sister passwords exhibits lower similarity with lower cosine similarity, LCS, and overlap score. Further, as shown in Figs. 1 and 2, for a user's password, the average 2-gram cosine (resp. LCS and overlap score) similarity score




2-gram cosine distribution LCS distribution Overlap distribution

**Fig. 2:** Measuring the similarity of the sister password pair chosen by the same user across different services. The similarity between sister passwords is significantly lower than that between users' passwords and popular passwords (Fig. 1), highlighting that using sister passwords is a more complex user behavior.

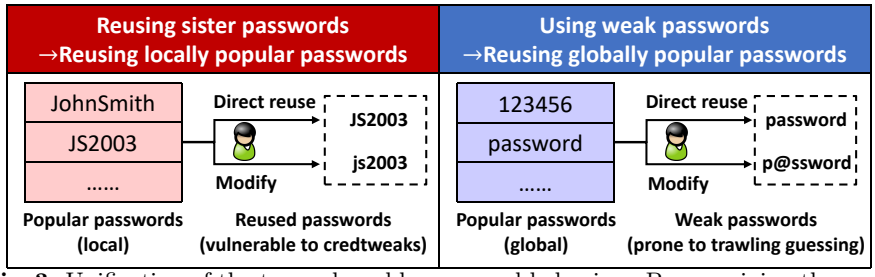



**Fig. 3:** Unification of the two vulnerable password behaviors. By perceiving the use of weak passwords and the use/modification of sister passwords as reusing globally and locally popular passwords, these two vulnerable behaviors can be seen as one.

between it and her sister password is 0.302 (resp. 3.09 and 1.95), while the number is 0.475 (resp. 3.68 and 7.91) between it and its most similar popular password. For instance, when the sister password `alice@hotmail.com` is modified to `123456alice`, the 2-gram cosine similarity score is 0.39; In contrast, when the popular password `123456` is modified to `123456alice`, the cosine similarity score is 0.65. Our results are statistically significant, as the results of Mann–Whitney U tests (which examine whether the distributions of two groups are different) lead to $p$-values$<10^{-3}$. Levenshtein and Manhattan distances show the same trend: half of the passwords differ by at most 5 edits from the corresponding popular passwords, strengthening that modified passwords are often similar to popular ones. More detailed results are shown in Appendix C.

**Summary**. Findings 1 and 2 lay empirical foundations for constructing a versatile framework for password strength evaluations (PSEs) via continual learning techniques. Firstly, both credential tweaking and trawling guessing attacks exploit reuse behaviors: As Fig. 3 shows, password strengths against both attacks can be expressed as a unified conditional probability whose priors are a sister password (for tweaking) and a popular password dictionary (for trawling), respectively. Secondly, the levels of complexity in users' vulnerable behaviors vary, so a PSE should simultaneously capture vulnerable user behaviors of varying complexity. This motivates us to employ continual learning techniques in PSEs.

## 4 VersaPSE: A reuse-based, multipurpose password strength evaluation framework

We employ continual learning techniques to build a generic framework named VersaPSE for evaluating password strengths against both *credential tweaking and trawling guessing attacks.* We first provide theories for proper continual learning, then construct our VersaPSE framework in a theory-guided approach.

### 4.1 Theories of continual learning across user behaviors

Continual learning allows a model to be trained on a series of related tasks, and achieve strong performance on both tasks simultaneously. At a high level, continual learning sequentially trains a model on multiple tasks to master the entire set, while enabling knowledge transfer between them [23, 73], and it has been widely used in domains like natural language processing [49, 54] and computer vision [39, 59]. *In essence, effective continual learning necessitates the proper sequencing of tasks and the application of parameter freezing to reconcile new knowledge acquisition with the preservation of prior learning [23, 43, 82].* Since we need to deal with two tasks, i.e., local and global reuse behaviors, we refer to them as *initial* and *subsequent* tasks based on their implementation order.

Inspired by machine learning studies [22, 25, 30, 47, 48], we explore the efficacy of continual learning in terms of generalization and forgetting risks. For theoretical clarity, we assume models are trained to reach optimality and denote the pre-trained and fine-tuned models as $\boldsymbol{\theta}_{\text{pt}}, \boldsymbol{\theta}_{\text{full}} \in \mathbb{R}^d$, respectively. In short, a generalization risk $\mathcal{R}_{\text{gen}}$ captures the ability of transferring knowledge from the initial to the subsequent tasks, and a forgetting risk $\mathcal{R}_{\text{forget}}$ measures initial-task degradation after fine-tuning for the subsequent task. We formalize these risks and provide guidelines for minimizing them as follows.

**Definition 1.** ***Generalization risk.*** *For a hypothesis $h_t : \mathcal{X} \to \mathcal{Y}$ and a bounded loss function $\ell_t : \mathcal{Y} \times \mathcal{Y} \to [0, b]$ for the subsequent task on the fine-tuned model $\boldsymbol{\theta}_{\text{full}}$, the generalization risk is the difference between the empirical loss $\hat{\mathcal{L}}_t(h_t) = \frac{1}{n}\sum_{i=1}^{n} \ell_t(h_t(x_i), y_i)$ and its expectation $\mathcal{L}_t(h_t) = \mathbb{E}_{\mathcal{D}_t}[\ell_t(h_t(x), y)]$, i.e.,*

$$\mathcal{R}_{gen}(h) = \mathbb{E}_{\mathcal{D}_t}\left[\ell_t(h_t(x), y)\right] - \frac{1}{n}\sum_{i=1}^{n} \ell_t(h_t(x_i), y_i), \tag{1}$$

*where $\mathcal{D}_t$ denotes the unknown data distribution over $\mathcal{X} \times \mathcal{Y}$ of the subsequent task, and $\{(x_i, y_i)\}_{i=1}^{n}$ denote $n$ samples drawn from $\mathcal{D}_t$. The generalization risk is upper-bounded by Rademacher complexity (see Lemma 1 of Appendix A).*
***Empirical forgetting risk*** *is the difference between the empirical task loss before and after fine-tuning the pre-trained model for the subsequent task, i.e.,*

$$\mathcal{R}_{forget}(s, t) = \frac{1}{n}\sum_{i=1}^{n} \big(\ell_s(h_t(x_i), y_i) - \ell_s(h_s(x), y)\big). \tag{2}$$

*where $\ell_s$ is the loss function for the initial task on the pre-trained model.*

We explain Definition 1. Eq. 1 defines generalization risk as the deviation of empirical loss from its expectation, and a smaller value indicates better generalization. Eq. 2 provides a computable empirical estimate of the forgetting risk for initial tasks with and without fine-tuning. We do not use the true expectation

of forgetting risk $\mathbb{E}_{\mathcal{D}_s}\left[\ell_s(h_t(x), y)\right] - \mathbb{E}_{\mathcal{D}_s}\left[\ell_s(h_s(x), y)\right]$ because it is intractable. Hence, unless otherwise noted, a forgetting risk refers to the empirical one.

The following Theorem 1 establishes upper bounds for generalization and forgetting risks, principally guiding initial and subsequent task selections.

**Theorem 1.** *Let $t_A \to t_B$ denote the case where the initial and subsequent tasks are $t_A$ and $t_B$, respectively. For the generalization risk, their upper bounds have $\mathcal{G}_{t_A \to t_B} \leq \mathcal{G}_{t_B \to t_A}$ if there are $\mathcal{H}_{t_A} \supseteq \mathcal{H}_{t_B}$ and $n_{t_A} \leq n_{t_B}$ for their hypothesis classes and sample sizes, respectively. For the forgetting risk, if the gradient and Hessian matrix of the initial-task loss function against the model parameter $\boldsymbol{\theta}$ satisfy $\|\nabla \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\| = 0$ and $\|\nabla^2 \mathcal{L}_s(\boldsymbol{\theta}) - \nabla^2 \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\| \leq \rho_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{pt}}\|$ for some $\rho_2$ in the vicinity of the pre-trained model $\boldsymbol{\theta}_{\mathrm{pt}}$, then there exists*

$$\mathcal{R}_{forget}(t_A, t_B) \leq \frac{1}{2}\Big(\lambda_{\max} \nabla^2 \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\Big) \|\boldsymbol{\delta}\|^2 \tag{3}$$

*where $\lambda_{\max}$ is the maximum eigenvalue of the Hessian matrix and $\boldsymbol{\delta} = \boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{pt}}$ is the model parameter update near $\boldsymbol{\theta}_{\mathrm{pt}}$.*

Theorem 1 reveals that this complex-to-simple continual learning can optimize the upper bounds of both generalization and forgetting risks. Firstly, for the generalization risk, the upper bound satisfies $\mathcal{G}_{t_A \to t_B} < \mathcal{G}_{t_B \to t_A}$ with a more complex $t_A$ task, indicating less overfitting for the subsequent task. Second, the empirical forgetting risk $\mathcal{R}_{forget}$ can also be reduced, as it scales with the magnitude of the parameter displacement $\|\boldsymbol{\delta}\|$ between the pre-trained and fine-tuned models. Pre-training on a more complex initial task places the model within a richer hypothesis class that approximates the simpler subsequent task, naturally constraining $\|\boldsymbol{\delta}\|$ in fine-tuning and limiting initial-task degradation [20,33,43,74,75].

We then present the overall layer-freezing strategy used for fine-tuning.

**Theorem 2.** *Let the pre-trained model $\boldsymbol{\theta}_{\mathrm{pt}}$ for the initial task $t_A$ be partitioned into frozen (F) and unfrozen (UF) subsets with the update $\boldsymbol{\delta}$ being $\boldsymbol{\delta}^{\mathrm{full}} = (\boldsymbol{\delta}_F^{\mathrm{full}}, \boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{full}})$ and $\boldsymbol{\delta}^{\mathrm{frz}} = (\mathbf{0}, \boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{frz}})$ for the subsequent task $t_B$. When using full fine-tuning and partial fine-tuning, the upper bound of the generalization risks has $\mathcal{G}^{\mathrm{frz}} < \mathcal{G}^{\mathrm{full}}$ if their hypothesis classes have $\mathcal{H}_{\mathrm{frz}} \subseteq \mathcal{H}_{\mathrm{full}}$, and the difference in the forgetting risks of the initial task has the following*

$$\begin{aligned}
&\mathcal{R}_{forget}^{\mathrm{full}}(t_A, t_B) - \mathcal{R}_{forget}^{\mathrm{frz}}(t_A, t_B) \\
&= \frac{1}{2}(\boldsymbol{\delta}_F^{\mathrm{full}})^\top H_{F,F} \boldsymbol{\delta}_F^{\mathrm{full}} + (\boldsymbol{\delta}_F^{\mathrm{full}})^\top H_{F,\mathrm{UF}} \boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{full}} + \frac{1}{2}\Big[(\boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{full}})^\top H_{\mathrm{UF},\mathrm{UF}} \boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{full}} - (\boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{frz}})^\top H_{\mathrm{UF},\mathrm{UF}} \boldsymbol{\delta}_{\mathrm{UF}}^{\mathrm{frz}}\Big],
\end{aligned} \tag{4}$$

*with the Hessian matrix $H_S = \nabla^2 \mathcal{L}_S(\boldsymbol{\theta}_{\mathrm{pt}}) = \begin{bmatrix} H_{F,F} & H_{F,\mathrm{UF}} \\ H_{\mathrm{UF},F} & H_{\mathrm{UF},\mathrm{UF}} \end{bmatrix}$.*

Theorem 2 shows the necessity of employing partial fine-tuning with frozen parameters. More specifically, to minimize the forgetting risk, frozen weights should strongly correlate with the initial task (i.e., large $H_{F,F}$ diagonals) and weakly correlate with the subsequent task (i.e., small $H_{F,\mathrm{UF}}$ entries). We will explicate the details on such parameter selection in the next section.

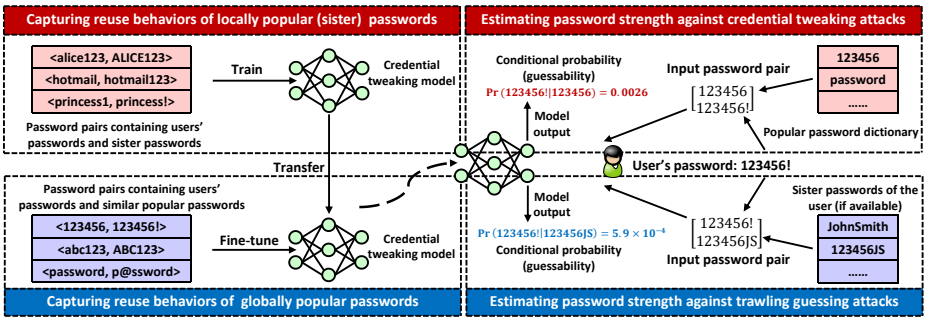



**Fig. 4:** The workflow of VersaPSE. In training, the model first captures reuse behaviors based on sister passwords (local reuse), then it is fine-tuned to capture reuse behaviors of popular passwords (global reuse). After training, VersaPSE outputs two conditional probabilities, representing password strength against credential tweaking or trawling guessing attacks.

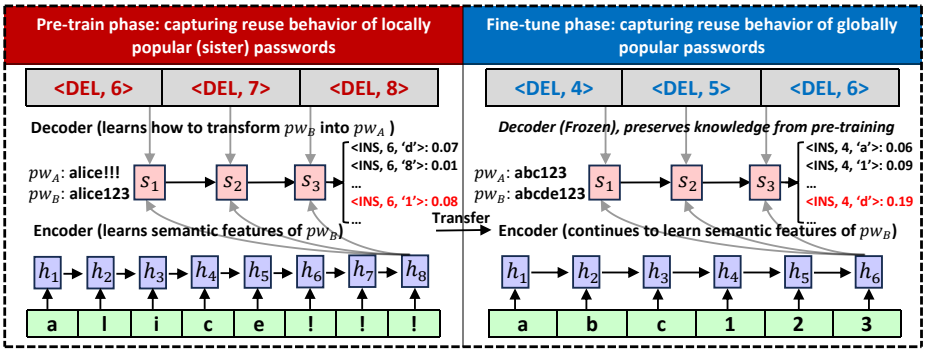



**Fig. 5:** An illustration on how to freeze parameters when our VersaPSE framework is instantiated on the first credential tweaking model Pass2Path [51].

## 4.2 Workflow of VersaPSE

Based on our theories, we now propose VersaPSE, which utilizes a credential tweaking model to measure password strength against both credential tweaking and trawling guessing attacks. At a high level, VersaPSE incorporates a credential tweaking model, which has undergone continual learning, and a popular password dictionary against trawling guessing attacks. As guided by Theorem 1, we treat the more complex task as the initial task (i.e., trained first), and the other as the subsequent task (see Fig. 4). Besides, since Finding 2 shows that local reuse behavior is more complex than global reuse, we choose local and global reuse behaviors as the initial and subsequent tasks, respectively.

In addition, following Theorem 2, we also freeze the model parameters relevant to capturing how users modify passwords, and fine-tune the parts that capture semantics of the source password (e.g., the user's sister password or popular ones). This strategy stems from Findings 1 and 2, as freezing parts related to more complex modification behaviors in the initial task provides better transferability for handling the subsequent task (i.e., the simpler global reuse).

We use VersaPSE instantiated with Pass2Path [51] as an example. As Fig. 5 shows, the encoder of Pass2Path learns semantics while the decoder outputs structural edit operations. Hence, when fine-tuning on global reuse behaviors, we freeze the decoder to preserve knowledge of users' more complex modification preferences and only update the encoder. As for parameter freezing choices for other models, the general principle is to freeze parameters related to capturing the modification/editing process during password reuse (see Appendix B).

To derive a training set for global reuse (i.e., reusing popular passwords) in VersaPSE, we first build a dictionary of the top-$10^3$ most frequent passwords. For every password with frequencies$\geq$100, we compute its 2-gram cosine similarity against these popular passwords, and then retain up to five popular passwords whose similarity exceeds 0.5, pairing each as $\langle ppw, pw \rangle$. Here, the frequency threshold of 100 reduces training time and prevents over-fitting to global reuse. We also investigated several different cosine similarity thresholds within the range of $[0.3, 0.7]$, and found only marginal differences, where a higher threshold modestly improves accuracy on local reuse but slightly degrades model performance on global reuse (i.e., slightly improved accuracy for targeted evaluation at a small cost on accuracy for untargeted strength evaluation). As for the construction of training sets for learning local reuse behavior (i.e., reusing sister passwords), we follow prior work [42, 70, 79], which matches sister passwords using email address (see Section 3.1), yielding password pairs with each pair containing two passwords of the same user.

After continual learning, the credential tweaking model can serve as a multipurpose PSM. Essentially, when given a sister/popular password (denoted as the base password), credential tweaking models output a series of variants in descending order of conditional probability, which represents the likelihood of a user's adoption of a password based on a given password (i.e., conditional probability). For example, when a user's password is `JohnSmith`, the conditional probability Pr(`johnsmith`|`JohnSmith`) represents the likelihood of the user modifying `JohnSmith` into `johnsmith`. *This conditional probability is then utilized as a representation of password strength under both credential tweaking and trawling guessing attacks.* For credential tweaking attacks, as Fig. 4 shows, the conditional probability represents the strength of the targeted password when the sister password is given, and the higher the probability, the more *vulnerable* the target password is.

For trawling guessing attacks, since using weak passwords can be seen as reusing popular passwords, credential tweaking models are also applicable. More concretely, as shown in Sec. 4.1, VersaPSE first matches the user's password with the most similar popular password by using 2-gram cosine similarity metric, with the popular password serving as the base password. The two passwords are input to the credential tweaking model, yielding a conditional probability. For instance, `password123` will be matched with the popular `password`, with the credential tweaking model outputting Pr(`password123`|`password`). Since trawling guessing exploits behaviors of reusing popular passwords (i.e., using weak passwords), the probability is a representation of password strength against trawling attacks.

**Table 2:** Credential tweaking attacks and password strength evaluation scenarios

| No. | Language (Subsequent)‡ | Training set (Initial)* † | Size (pairs) | Training set (Attack)† | Size (pairs) | Test set* | Size (pairs) | Cracked |
|---|---|---|---|---|---|---|---|---|
| #1 | Chinese (Tianya) | 126-Dodonew | 49032 | Tianya-Dodonew | 445514 | Tianya-Taobao | 455847 | 11.91% |
| #2 | | Tianya-Dodonew | 445514 | 126-Dodonew | 49032 | 126-CSDN | 57213 | 34.23% |
| #3 | | Tianya-Dodonew | 445514 | CSDN-Dodonew | 160199 | CSDN-126 | 57213 | 48.30% |
| #4 | | CSDN-Dodonew | 160199 | Tianya-Dodonew | 445514 | Tianya-CSDN | 552263 | 35.23% |
| #5 | English (Rockyou) | LinkedIn-Twitter | 300370 | 000webhost-Twitter | 146945 | 000webhost-LinkedIn | 214346 | 24.36% |
| #6 | | 000webhost-Twitter | 146945 | Twitter-LinkedIn | 300370 | Twitter-000webhost | 295186 | 17.04% |
| #7 | | LinkedIn-Mathway | 111637 | LinkedIn-Twitter | 111637 | LinkedIn-000webhost | 214346 | 18.26% |
| #8 | | Twitter-LinkedIn | 300370 | LinkedIn-Twitter | 300370 | LinkedIn-MathWay | 111637 | 21.79% |

* Datasets in these columns are password pairs matched via email address, following prior work [42,71,79]; e.g., in "126-Dodonew", a user's 126 password serves as the sister password to evaluate/attack her Dodonew password. When an email is associated with multiple passwords in the same dataset, a random one is chosen.
† Training sets (Initial) are used to train the base model of VersaPSE and password strength meters (PSMs). Training sets (Attack) are used to train credential tweaking models.
‡ Subsequent indicates password datasets used for the subsequent task in continual learning for VersaPSE. Although Tianya appears in both the training and test sets, no data leaks occur, as the training set is used as a general dictionary, and the test set contains matched password pairs from two datasets.

**Remark.** In essence, our VersaPSE improves upon existing PSM designs (e.g., KNN-PSM [42], PR-PSM [79], and BERT-PSM [80]) in three key respects. First, it unifies local and global reuse and can evaluate password strengths against both credential tweaking and trawling guessing attacks, whereas prior PSMs often treat them separately. Second, it computes conditional probabilities instead of enumerating variants or measuring symmetric similarity, capturing users' password modification behaviors and measuring risks in a continuous approach. Third, it uses continual learning to transfer from local to global reuse, selectively freezing parameters to retain prior knowledge while avoiding per-scenario retraining, allowing VersaPSE to adapt efficiently to new reuse distributions.

## 5 Credential tweaking attack evaluations

### 5.1 Experimental setups

To simulate real-world credential tweaking attacks, we utilize password datasets in Table 1 and set up eight attack scenarios (see Table 2) where attackers use six credential tweaking models (i.e., KNNGuess [42], PointerGuess [79], Pass2Edit [71], PassBERT [80], Pass2Path [51], and TarGuess-II [70]), as well as popular password dictionaries in attacks. To begin with, since password strengths are heavily impacted by users' languages [69], we first partition datasets into Chinese and English ones. We introduce the eight attack scenario settings as follows.

Scenario #1 simulates the case where an attacker obtains a user's password for a service and tries to attack another similar service (e.g., both Dodonew and Taobao are Chinese e-commerce websites). Scenarios #2, #4, #6, #7 simulate the case where an attacker acquires passwords from sites with weaker creation policies and attacks accounts of sites with stronger ones (e.g., 000webhost has stronger creation policies than Twitter). Conversely, Scenarios #3, #5, #8 simulate the case where attackers try to crack accounts with weaker creation policies when obtaining strong sister passwords. As attackers also mix popular passwords in credential tweaking attacks to enhance attack efficiency [42,70,71,79], we use the top-$10^3$ most popular passwords in each test set as attackers' popular password dictionaries as in [31,68]. In each scenario, the training sets of PSMs and attackers differ, since services are unlikely to know the attackers' full knowledge.

We now present how we evaluate the accuracy of PSMs against credential tweaking attacks. First, in attacks, we use credential tweaking models to determine whether a password is secure: A password is secure if it can neither be cracked by the above six mainstream credential tweaking models in $10^3$ guesses (see [42,51,70,71,79,80]), nor belong to the top-$10^3$ popular passwords [31,68]. Accordingly, in password strength evaluations, we integrate Zxcvbn-PSM [77], which is a widely-used industrial PSM for detecting weak passwords [31,68], with reuse-based PSMs as recommended in [51,79] to assist detecting weak passwords: When the guess number of a password calculated by Zxcvbn-PSM [77] is lower than $10^3$, the password is marked as insecure regardless of its output strength from reuse-based PSMs. We do not integrate Zxcvbn-PSM [77] with our VersaPSE, since it is designed to also measure password strengths against trawling guessing attacks (see Appendix B for more detailed setups).

### 5.2 Evaluation metrics

As revealed in [51,52], a PSE against credential tweaking attacks can be formulated as a binary classification task that categorizes passwords as secure or insecure with a given sister password. To properly evaluate performance on highly imbalanced datasets, we use the balanced accuracy (BA) metric as in [18,32,40] to capture the mean of class-specific accuracy, ensuring that model performance is not masked by the imbalanced impacts of the major class. Formally, let $p$ and $n$ be the numbers of positive (insecure) and negative (secure) test samples, and $p'$, $n'$ be the numbers of correctly classified positives and negatives. Here, a positive sample is one that can be cracked within $10^3$ guesses or belongs to the top-$10^3$ popular passwords (see Sec. 5.1). Thus, BA is then defined as

$$BA = (p'/p + n'/n)/2 = (TPR + TNR)/2, \tag{5}$$

where $TPR$ and $TNR$ denote the true positive and true negative rates in evaluating password strengths, respectively. We avoid using the accuracy metric (i.e., the proportion of passwords that are correctly identified as secure/insecure), as it yields misleadingly high scores for imbalanced classes [36, 40]. For instance, in Scenario #1, only 11.91% of password pairs are cracked (i.e., insecure), but a pseudo-classifier labeling all passwords as secure yields an unreasonably high accuracy of 0.88, obscuring actual efficacy. We also consider AUC and F1-Score metrics [36, 40], and their results are largely consistent (see Appendix D).

### 5.3 Evaluation results of credential tweaking attacks

Table 3 shows the experiment results of reuse-based PSMs. Firstly, for Vec-PPSM [51], BERT-PSM [80] and KNN-PSM [42], the results in the brackets (e.g., [0.827]) represent Zxcvbn-PSM-assisted [77] results. For the five multipurpose PSMs using our VersaPSE, we calculate the relative improvements compared with existing reuse-based PSMs using $(BA_{ours}–BA_{base})/BA_{base}$, and they are shown in parentheses, e.g., (+11.8%∼+72.2%). These results demonstrate the effectiveness of our VersaPSE framework in deriving multipurpose PSMs that accurately evaluate password strength under credential tweaking attack. More

**Table 3:** Balanced accuracy for credential tweaking attack scenarios†

| Scenario | VersaPSE (KNNG.)* | VersaPSE (Pass2E.)* | VersaPSE (PointerG.)* | VersaPSE (PassBERT)* | VersaPSE (Pass2P.)* | KNN-PSM | PR-PSM | BERT-PSM | Vec-PPSM |
|---|---|---|---|---|---|---|---|---|---|
| #1 | 0.930 (+11.8%∼ +72.2%) | 0.918 (+10.3%∼ +70.0%) | 0.867 (+4.2%∼ +60.5%) | 0.812 (-2.5%∼ +50.3%) | 0.906 (+8.9%∼ +67.8%) | 0.695 [0.832] | 0.719 | 0.540 [0.679] | 0.719 [0.767] |
| #2 | 0.883 (+4.0%∼ +45.7%) | 0.892 (+5.1%∼ +47.2%) | 0.864 (+1.8%∼ +42.6%) | 0.812 (-4.3%∼ +34.1%) | 0.875 (+3.1%∼ +44.4%) | 0.606 [0.827] | 0.849 | 0.636 [0.840] | 0.623 [0.732] |
| #3 | 0.954 (+1.1%∼ +76.3%) | 0.951 (+0.8%∼ +75.8%) | 0.938 (-0.5%∼ +73.4%) | 0.883 (-6.3%∼ +63.3%) | 0.909 (-3.6%∼ +68.1%) | 0.541 [0.688] | 0.943 | 0.786 [0.856] | 0.667 [0.732] |
| #4 | 0.890 (+9.3%∼ +57.5%) | 0.881 (+8.3%∼ +56.0%) | 0.861 (+5.8%∼ +52.4%) | 0.803 (-1.4%∼ +42.1%) | 0.864 (+6.1%∼ +52.9%) | 0.601 [0.814] | 0.799 | 0.565 [0.788] | 0.620 [0.709] |
| #5 | 0.950 (+4.5%∼ +65.5%) | 0.941 (+3.5%∼ +63.9%) | 0.927 (+2.0%∼ +61.5%) | 0.886 (-2.5%∼ +54.4%) | 0.916 (+0.7%∼ +59.5%) | 0.574 [0.710] | 0.909 | 0.814 [0.886] | 0.680 [0.760] |
| #6 | 0.925 (+4.0%∼ +40.6%) | 0.895 (+0.6%∼ +36.1%) | 0.879 (-1.3%∼ +33.6%) | 0.852 (-4.3%∼ +29.5%) | 0.896 (+0.7%∼ +36.2%) | 0.803 [0.890] | 0.691 | 0.658 [0.754] | 0.705 [0.759] |
| #7 | 0.926 (+4.2%∼ +46.6%) | 0.901 (+1.4%∼ +42.6%) | 0.865 (-2.7%∼ +36.8%) | 0.869 (-2.2%∼ +37.6%) | 0.900 (+1.2%∼ +42.3%) | 0.830 [0.889] | 0.673 | 0.632 [0.699] | 0.722 [0.753] |
| #8 | 0.891 (+4.5%∼ +35.9%) | 0.869 (+1.8%∼ +32.4%) | 0.860 (+0.9%∼ +31.2%) | 0.833 (-2.3%∼ +27.0%) | 0.858 (+0.6%∼ +30.8%) | 0.728 [0.853] | 0.819 | 0.749 [0.844] | 0.656 [0.741] |
| Average | 0.919 (+13.0%∼ +36.7%) | 0.906 (+11.4%∼ +34.8%) | 0.883 (+8.6%∼ +31.3%) | 0.844 (+3.8%∼ +25.6%) | 0.891 (+9.5%∼ +32.5%) | 0.672 [0.813] | 0.800 | 0.673 [0.793] | 0.674 [0.744] |

† Results in green denote the respective multipurpose PSM derived using VersaPSE outperform all prior reuse-based PSMs in the corresponding scenario. Results in parentheses, e.g., (+11.8%∼+72.2%) are the improvements of VersaPSE instantiated PSMs compared with leading reuse-based PSMs, calculated as $RI{=}(BA_{ours}{-}BA_{base})/BA_{base}$. Results in brackets, e.g., [0.688] denote the accuracy of leading reuse-based PSMs integrated with Zxcvbn-PSM [77].

* VersaPSE denotes PSMs instantiated under VersaPSE with credential tweaking models in brackets. For example, VersaPSE (KNNG.) is the PSM derived from KNNGuess [42].

concretely, all five multipurpose PSMs derived from their credential tweaking models via our VersaPSE outperform their original PSMs at detecting weak passwords. We note that VersaPSE instantiated with PassBERT only marginally surpasses existing PSMs. This can be attributed to PassBERT's design limitation (see Section 5.2.1 of [80]), resulting in its inability to cover highly complex reuse behaviors (e.g., those that involve more than 3 character insertions).

Secondly, compared with enumeration-based KNN-PSM [42], PR-PSM [79] and BERT-PSM [80], PSMs derived via VersaPSE achieve 17.2%∼36.6% higher balanced accuracy. This addresses the inherent drawbacks of enumeration-based PSMs, where only a small proportion of insecure sister password variants is considered. For example, `lxm1982` remains uncracked by PassBERT [80] when `lxmlxm` is leaked, but it is insecure as PointerGuess [79] can crack it within $10^3$ guesses. Instead, VersaPSE derives the conditional probability of the user's password, achieving a more fine-grained strength evaluation. Moreover, VersaPSE (KNNGuess/PointerGuess/PassBERT) outperform their enumeration-based KNN-PSM, PR-PSM, and BERT-PSM by 0.160, 0.099, and 0.171 in terms of the balanced accuracy, respectively. This indicates that our VersaPSE provides a more accurate technical route for designing PSMs.

Thirdly, PSMs derived from VersaPSE also demonstrate superior performance with 25.3% ∼36.4% higher balanced accuracy than the similarity-based Vec-PPSM [51]. This overcomes the drawback of straightforwardly using the similarity score (e.g., embedding similarity) in strength evaluations. For example, the

embedding similarity is *symmetric*, whereas reuse behaviors largely depend on the base password the user reuses. As a result, the secure password `barrybarry` reused from `barryhappy` is deemed insecure by Vec-PPSM. In contrast, VersaPSE's conditional probability captures reuse behaviors exploited by credential tweaking attacks, and the results show that our conditional probability is more effective than the embedding similarity metric [15] used in [51].

Credential tweaking attacks also exploit users' behaviors of using weak passwords [70, 71, 79]. This leads to a rise in balanced accuracy when existing reuse-based PSMs are Zxcvbn-PSM-assisted, since Zxcvbn-PSM identifies weak passwords [77] rather than relying solely on a given sister password. For example, the sister password `Alice2024!` is unlikely to give attackers an advantage in cracking `abc123`, although `abc123` is weak and can be cracked by credential tweaking attacks. While other reuse-based PSMs resort to trawling-based PSMs to mitigate this threat, VersaPSE handles this on its own, as it captures how users edit (or directly reuse) popular passwords to create new ones. It can be seen that VersaPSE still greatly outperforms (3.8%∼36.7% higher balanced accuracy) all other Zxcvbn-PSM-assisted reuse-based PSMs, showing that VersaPSE can still effectively identify weak passwords. To further demonstrate this, we also explore VersaPSE's capability in resisting trawling guessing attacks, as we show below.

## 6 Trawling guessing attack evaluations

We now focus on online evaluation of password strength against online trawling guessing attacks, where the attackers attempt to crack accounts online without prior knowledge of users' sister passwords or any personally identifiable information. In this scenario, the best an attacker can do is to log in with popular passwords (e.g., generated by a guessing dictionary) in descending order of probability [68]. This naturally contributes to larger cracked proportions than trying less popular passwords. Such threats are also practical, as many services have leaked their password files more than once (e.g., Twitter [7] and Yahoo [4]), and different services may share the same popular passwords (e.g., `password123`).

To fairly measure PSM accuracy against trawling guessing attacks where no sister passwords are needed, we compare the performance of multipurpose PSMs under our VersaPSE framework without sister passwords, along with mainstream trawling-based PSMs, including four from academia and three from industry.

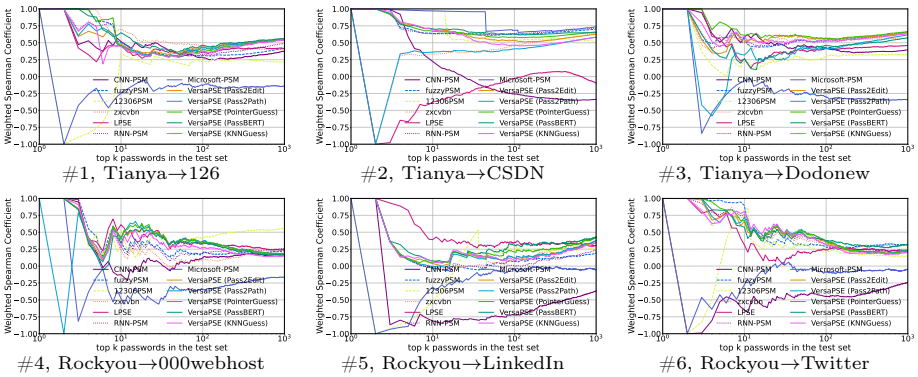
### 6.1 Experimental setups

As Table 4 shows, we utilize 12 datasets for our experiments. For VersaPSE, the training process is the same as that of the targeted password strength evaluation experiments. to ensure that VersaPSE against both credential tweaking and trawling attack scenarios can

**Table 4:** Trawling evaluation scenarios

| # | Language (Training set A*) | Training set B * | Training set C* | Test set |
|---|---|---|---|---|
| 1 | Chinese (Tianya) | Weibo | Tianya-Dodonew | 126 |
| 2 | | Dodonew | Tianya-Dodonew | CSDN |
| 3 | | Taobao | CSDN-126 | Dodonew |
| 4 | English (Rockyou) | Phpbb | Twitter-LinkedIn | 000webhost |
| 5 | | Twitter | 000webhost-Twitter | LinkedIn |
| 6 | | 000webhost | LinkedIn-MathWay | Twitter |

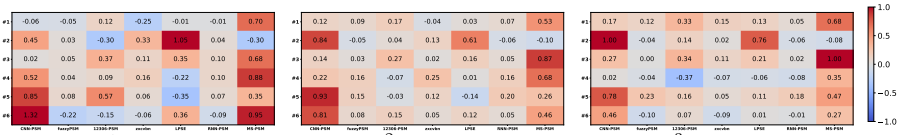
* Training set A is used to train PSMs that require a training set, as well as for the continual learning of VersaPSE. Training set B is used by fuzzyPSM [67], as it requires two training sets. Training set C contains sister password pairs and is used by VersaPSE before continual learning.




#1, Tianya→126 #2, Tianya→CSDN #3, Tianya→Dodonew

#4, Rockyou→000webhost #5, Rockyou→LinkedIn #6, Rockyou→Twitter

**Fig. 6:** Results for trawling password strength evaluation show that the 5 multipurpose PSMs derived under VersaPSE are among the best-performing PSMs, demonstrating the strong capability of the VersaPSE framework in adopting credential tweaking models to defend against trawling attacks.



Rank= 10 for each scenario. Rank= $10^2$ for each scenario. Rank= $10^3$ for each scenario.

**Fig. 7:** Heatmap for trawling evaluation results at different benchmark password ranks. Values in the heatmap indicate the increment (negative if decrement) achieved by VersaPSE (PointerGuess) over the corresponding baseline PSM. VersaPSE (PointerGuess) is generally the best performing.

have consistent performance. As for fuzzyPSM [67], besides the training set for capturing password semantics, it requires another training set for dataset-level reuse behaviors. Other leading trawling-based PSMs either require a single training set or none. For those that require a single training set, when evaluating Chinese users' passwords, we align with prior studies [68, 69], and use Tianya and Rockyou for Chinese and English passwords, respectively.

### 6.2 Evaluation metrics

As in prior studies [31, 67, 68], we mainly use the weighted Spearman correlation coefficient ($WSpearman$) metric to measure how a practical PSM deviates from the ideal PSM [67] that can capture true (but often unknown to security administrators) password probabilities. To begin with, an ideal PSM satisfies

$$M(pw) = p_{pw}, \forall pw \in \Gamma, \tag{6}$$

where $p_{pw}$ is the probability of password $pw$ from the true password distribution $\chi$. Since passwords are Zipf-distributed [66], probabilities of very unpopular passwords are unlikely to be accurately approximated by their frequencies [66], so we only consider passwords whose frequencies are above 10 as in [31, 68].

Next, we use $WSpearman$ to compare the results output by a practical PSM and the ideal PSM. Generally, the more similar the results are, the more accurate the PSM is. Formally, $WSpearman$ is defined as

$$WSpearman = \frac{\sum_{i=1}^{n} [w_i (x_i - \bar{x}) (y_i - \bar{y})]}{\sqrt{\sum_{i=1}^{n} [w_i (x_i - \bar{x})^2] \sum_{i=1}^{n} [w_i (y_i - \bar{y})^2]}}, \tag{7}$$

where $x_i$ and $y_i$ are the $i$-th elements of $X$ and $Y$, the weighted rank vectors for outputs of ideal PSM and the tested PSM, respectively, with $w_i$ being the corresponding weight that equals the $i$-th password's frequency. $\overline{x}$ and $\overline{y}$ are weighted means of $X$ and $Y$. $WSpearman$ ranges from $-1$ to 1, and a higher value indicates that the outputs of ideal PSM and the tested PSM are more similar, showing that the tested PSM is more accurate.

As $WSpearman$ outputs a single value for a password rank, we further utilize a $WSpearman$ curve to show how PSMs perform as the rank of passwords changes for more holistic views. We record discrete $WSpearman$ values at benchmark password ranks and calculate the increment (which is calculated as $WSpearman_{VersaPSE} - WSpearman_{otherPSM}$) for multipurpose PSMs derived via VersaPSE over other PSMs. As shown in Fig. 7, we choose VersaPSE (PointerGuess) as an example, and do not present heatmaps for other multipurpose PSMs here due to the page limit. This relative $WSpearman$ at different benchmark values provides more intuitive interpretations for evaluation results.

### 6.3 Evaluation results of trawling guessing attacks

Figs. 6 and 7 show the experiment results. Compared with entropy-based 12306-PSM [8] and Microsoft-PSM [9], all multipurpose PSMs under the VersaPSE framework perform significantly better with average increments of 0.07∼0.48 on all ranks, which is expected since they represent the average performance of industry-designed PSMs [68]. The pattern-based Zxcvbn-PSM [77] is often referred to as the most promising industry-designed PSM [31, 68], and our multipurpose PSMs achieve average improvements of 0.02∼0.06 in terms of the $WSpearman$ score. This further reveals the strong capability of the multipurpose VersaPSE, as trawling-based PSMs often require the assistance of Zxcvbn-PSM in practice for a trawling password strength evaluation (e.g., Vec-PPSM [51] and PR-PSM [79]), and our multipurpose PSMs derived under VersaPSE outperform Zxcvbn-PSM [77] even in trawling password strength evaluation scenarios.

When we compare multipurpose PSMs derived under VersaPSE with similarity-based or probabilistic PSMs (LPSE [34] and fuzzyPSM [67], resp.), VersaPSE is also highly competitive. Specifically, our multipurpose PSMs outperform LPSE by 0.45∼0.48 in $WSpearman$ at all three ranks. This is because LPSE utilizes an ideally strong password vector and similarity metrics to estimate password strength. However, users do not choose their password by editing a high-entropy password, while VersaPSE captures users' behaviors of editing/reusing popular passwords. In comparison with fuzzyPSM [67], the best-performing trawling-based PSM according to [31, 68], multipurpose PSM derived from PointerGuess under VersaPSE achieve average increases of 0.03 on all ranks. Though fuzzyPSM also captures population-wide reuse behaviors, it is based on statistical methods and thus lacks the generalization ability of the deep-learning-based VersaPSE.

CNN-PSM [53] and RNN-PSM [45] are so far the most promising deep-learning-based PSMs. The 5 multipurpose PSMs derived using VersaPSE out-

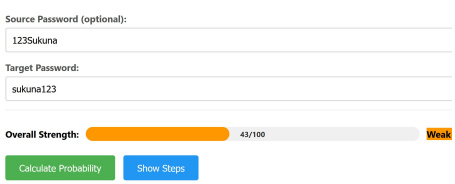



**Fig. 8:** Interface of VersaPSE implementation.

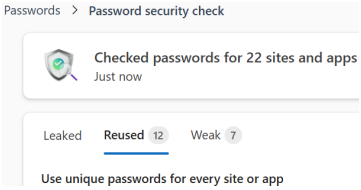



**Fig. 9:** Microsoft password manager [9].

perform CNN-PSM and RNN-PSM by 0.04∼0.49 at all benchmarks on average. This highlights the effectiveness of VersaPSE since it targets password reuse behaviors that neither CNN-PSM nor RNN-PSM considers, demonstrating the necessity to take password reuse behaviors into consideration when evaluating password strength. Furthermore, as VersaPSE is also based on deep learning, the results show that VersaPSE takes one step further towards introducing deep learning into trawling password strength evaluations.

# 7 Discussions

## 7.1 PSM deployability

Since passwords are users' private data, PSMs need to be sufficiently lightweight in terms of computation and storage to be deployed on the client side. To show this, we measure the computation times and storage costs of multipurpose PSMs derived from 5 foremost credential tweaking models on a laptop with an Intel Core i9-14900HX CPU, and compare them with existing KNN-PSM [42], PR-PSM [79] and BERT-PSM [80]. The results are shown in Table 5. First, all multipurpose PSMs under VersaPSE achieve sub-second computation times, while prior PR-PSM [79] and BERT-PSM [80] require 206∼1561 ms per password, which is noticeably slower than our multipurpose PSMs. The key reason is that VersaPSE directly computes conditional probabilities, avoiding computationally intensive enumerations in prior methods. Second, storage sizes depend on specific credential tweaking models used to derive PSMs, and they range from 8.86MB to 188.3MB. Since three out of five existing models occupy less than 20MB, we believe mainstream PSMs integrated with VersaPSE can be readily deployed on the client side in terms of storage. As a result, VersaPSE yields more lightweight PSMs than leading methods in terms of computation and storage. As for the training time, it is model-dependent and does not interfere with the PSM's deployment. Typically, under VersaPSE, a credential tweaking model (e.g., PointerGuess [79] and Pass2Edit [71]) can be trained in one hour on an NVIDIA RTX 4090 GPU.

**Table 5:** Deployability evaluation results

| Model type | Storage | Time |
|---|---|---|
| VersaPSE-KNNGuess | 188.3MB | 41.87ms |
| VersaPSE-Pass2Edit | 8.86MB | 24.82ms |
| VersaPSE-PointerGuess | 2.26MB | 15.99ms |
| VersaPSE-PassBERT | 81.2MB | 107.5ms |
| VersaPSE-Pass2Path | 14.8MB | 15.48ms |
| PR-PSM [79] | 2.49MB | 1561ms |
| BERT-PSM [80] | 81.2MB | 206.6ms |
| KNN-PSM [42] | 188.3MB | 1450ms |

We also provide a proof-of-concept implementation of our VersaPSE as a client-side browser extension. More specifically, we choose the lightweight and fastest VersaPSE (PointerGuess), and export the model to the ONNX format, which supports inference through the built-in WebAssembly with a CPU. The user interface is shown in Fig. 8, and the workflow goes as follows. VersaPSE

(PointerGuess) first pairs the targeted password (to be evaluated) with the sister password (e.g., provided by the user or within password managers) if available. The targeted password will also be paired with the most similar popular (e.g., a top-$10^3$) password from a password dictionary (e.g., Tianya for Chinese users). For each pair, the model calculates the conditional probability of the targeted password given the paired password, and the maximum number is taken as the password strength against guessing attacks.

Further, to facilitate users' understanding of password strengths, we also convert the conditional probability into a more intuitive strength score. To do this, we partition the test passwords into cracked and uncracked groups. For each group, we evaluate all passwords using VersaPSE to obtain their conditional probability mass $\Pr(\cdot)$. We further denote $\mathrm{CCDF}_{\mathrm{crk}}(x)$ (resp. $\mathrm{CCDF}_{\mathrm{uncrk}}(x)$) as the probability mass of passwords whose probabilities (given by VersaPSE) are smaller than $x$ in the cracked (resp. uncracked) group. In this way, the score is

$$\mathrm{Score}(x) = \frac{\mathrm{CCDF}_{\mathrm{uncrk}}(x)\Pr(uncrk) - \mathrm{CCDF}_{\mathrm{crk}}(x)\Pr(crk)}{\mathrm{CCDF}_{\mathrm{uncrk}}(x)\Pr(uncrk) + \mathrm{CCDF}_{\mathrm{crk}}(x)\Pr(crk)} \tag{8}$$

This score ranges from -1 to 1, with positive values indicating higher confidence that the password will not be cracked. We then linearly map this score to a more intuitive scale of 0 to 100, enabling a strength bar in Fig. 8. The resulting score provides an interpretable password strength metric for end users, where higher values indicate stronger passwords. In addition, privacy concerns may arise because the score is computed from conditional probabilities extracted from the test set. To reduce privacy leakage, we can further employ differential privacy techniques (e.g., [13, 26]) on conditional probabilities.

### 7.2 Further applications

The first application is integrating VersaPSE with a client-side password manager that stores a user's passwords in an encrypted vault. A password manager can obtain a user's plaintext passwords once the encrypted vault is decrypted with the correct master password. Deployed password managers (e.g., the one built into Microsoft Edge browser [9], see Fig. 9) mainly focus on detecting weak and *exact* reused passwords (e.g., via C3 services like HIBP [11]), leaving unaddressed the risk of credential tweaking attacks. VersaPSE can bridge this gap by serving as a general-purpose PSM against online guessing attacks for password managers. In particular, when the vault is correctly decrypted, VersaPSE can accurately evaluate the strength of a password if one or multiple other passwords stored in the manager are leaked. This can nudge a user to better reset her passwords when one or more similar (e.g., sister) passwords are leaked.

Additionally, as revealed in Nisenoff et al.'s two-decade retrospective study on password reuse [50], security administrators are advised to obtain leaked password files and apply tweaking rules (e.g., capitalization and adding special symbols) to check whether leaked passwords are reused. In this case, by accessing users' previously leaked plaintext passwords, VersaPSE can assist administrators in determining whether a user's password is vulnerable on the server-side, prior to hashing the password. Moreover, VersaPSE can be deployed in browsers and

webpages to evaluate password strengths on the client-side with low overhead and storage cost, serving as a reuse-based PSM resisting trawling attacks.

### 7.3 Limitations and future work

**Accuracy in defending against offline guessing attacks.** In this work, VersaPSE focuses on defending against online password guessing attacks and demonstrates the feasibility of applying continual learning across two different password strength evaluation scenarios. For future work, VersaPSE might be extended to estimating password strength against offline guessing attacks by following the experimental setups in [64,68]. This is potentially feasible, since offline guessing attacks exploit the vulnerable behaviors of using weak and medium-strength passwords, we can extend VersaPSE's fine-tuning phase to the reuse behavior of modifying popular passwords to medium-strength passwords.
**User-friendly feedback and interface design.** The browser implementation of VersaPSE in Sec. 7.1 aims to demonstrate the feasibility of VersaPSE by directly deploying it on the client side. For strength feedback, we opted for a probabilistic approach to derive a strength score, rather than showing a probability calculated by VersaPSE. Nevertheless, how to nudge users towards more secure passwords *in the context of a credential tweaking attack*, such as providing actionable modification suggestions, remains an open question. A straw proposal is to incorporate explainable heuristics with VersaPSE [63], or conduct a user study to refine the user interface of our meter. We leave them as future work.

## 8 Conclusion

In this paper, we, *for the first time*, have adopted a principled approach to integrate continual learning techniques into password strength evaluations (PSEs). We have proposed VersaPSE, a versatile reuse-based framework to transform credential tweaking models into multipurpose password strength meters (PSMs) that can accurately evaluate password strengths against both credential tweaking and online trawling guessing attacks. Using VersaPSE, we have derived five multipurpose PSMs from the leading credential tweaking models, and demonstrated their superiority in evaluating password strength against both credential tweaking and trawling guessing attacks. We believe this work introduces an important technical route towards designing accurate and deployable PSMs in practice.

## Acknowledgment

The authors are grateful to the anonymous reviewers for their invaluable comments. Zhenduo Hou is the corresponding author. The authors would like to thank Yuanwei Ji for the helpful discussion and proofreading of the camera-ready version. This research was in part supported by the Natural Science Foundation of Tianjin, China under Grant No. 25JCJQJC00230. Partial code (with ethics preserved) of our artifact is publicly available at https://github.com/FelicRivarez/VersaPSE_code .

## References


1. Bitwarden: How strong is your password? https://bitwarden.com/password-strength/
2. What is zxcvbn.min.js in wordpress? https://webtrainingwheels.com/zxcvbn-wordpress/
3. Zxcvbn: realistic password strength estimation. https://dropbox.tech/security/zxcvbn-realistic-password-strength-estimation (Apr 2012)
4. Yahoo data breaches. https://en.wikipedia.org/wiki/Yahoo_data_breaches (2016)
5. The 2021 psychology of passwords report. https://www.lastpass.com/resources/ebook/psychology-of-passwords-2021 (2021)
6. The 2022 psychology of passwords report. https://www.lastpass.com/-/media/3c627ed089e84bc39ca2bf6bf1d7cdec.pdf (2022)
7. Twitter data breaches: Full timeline through 2023. https://firewalltimes.com/twitter-data-breach-timeline/ (2023)
8. 12306 registration page. https://kyfw.12306.cn/otn/regist/init (Aug 2024)
9. Edge password health indicator. https://support.microsoft.com/en-US/edge/password-health-indicator (Aug 2024)
10. Microsoft digital defense report 2025: Lighting the path to a secure future (June 2025), https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/msc/documents/presentations/CSR/Microsoft-Digital-Defense-Report-2025.pdf#page=1
11. Pwned passwords: Check if your password has appeared in known data breaches. https://haveibeenpwned.com/Passwords (2025)
12. World Password Day 2025 Survey: 72% of Gen Z reuse passwords (May 2025), https://bitwarden.com/resources/world-password-day/
13. Blocki, J., Datta, A., Bonneau, J.: Differentially private password frequency lists. In: Proc. NDSS 2016
14. Blocki, J., Liu, P.: Towards a rigorous statistical analysis of empirical password datasets. In: Proc. IEEE S&P 2023. pp. 607–625
15. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguistics **5**, 135–146 (2017)
16. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: Proc. IEEE S&P 2012. pp. 538–552
17. Bonneau, J., Herley, C., van Oorschot, P., Stajano, F.: Passwords and the evolution of imperfect authentication. Commun. ACM **58**(7), 78–87 (2015)
18. Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: Proc. ICPR 2010. pp. 3121–3124
19. Cheng, H., Li, W., Wang, P., Chu, C.H., Liang, K.: Incrementally updateable honey password vaults. In: USENIX SEC 2021. pp. 857–874
20. Cho, M., Park, J., Lee, S., Sung, Y.: Hard tasks first: Multi-task reinforcement learning through task scheduling. In: Proc. ICML 2024
21. Cimpanu, C.: Hacker leaks 15 million records from Tokopedia, Indonesia's largest online store (May 2020), https://www.zdnet.com/article/hacker-leaks-15-million-records-from-tokopedia-indonesias-largest-online-store/
22. Cortes, C., Mohri, M., Rostamizadeh, A.: Generalization bounds for learning kernels. In: Proc. ICML 2010. pp. 247–254 (2010)
23. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in

classification tasks. IEEE Trans. Pattern Anal. Machine Intell. **44**(7), 3366–3385 (2021)
24. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. NAACL 2019
25. Doan, T., Bennani, M.A., Mazoure, B., Rabusseau, G., Alquier, P.: A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In: Proc. AISTATS. pp. 1072–1080 (2021)
26. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**, 211–407 (2014)
27. Eitel, B.: Will passwords become a thing of the past? (Mar 2023), https://www.idsalliance.org/will-passwords-become-a-thing-of-the-past/
28. Galbally, J., Coisel, I., Sanchez, I.: A new multimodal approach for password strength estimation - Part I: Theory and algorithms. IEEE Trans. Inf. Forensics Secur. **12**(12), 2829–2844 (2017)
29. Gatlan, S.: Hacker leaks full database of 77 million Nitro PDF user records (Jan 2021), https://www.bleepingcomputer.com/news/security/hacker-leaks-full-database-of-77-million-nitro-pdf-user-records/
30. Goldfarb, D., Evron, I., Weinberger, N., Soudry, D., Hand, P.: The joint effect of task similarity and overparameterization on catastrophic forgetting—an analytical model. In: Proc. ICLR 2024
31. Golla, M., Dürmuth, M.: On the accuracy of password strength meters. In: Proc. ACM CCS 2018. pp. 1567–1582
32. Grandini, M., Bagli, E., Visani, G.: Metrics for multi-class classification: an overview. ArXiv **abs/2008.05756** (2020)
33. Guo, M., Haque, A., Huang, D.A., Yeung, S., Fei-Fei, L.: Dynamic task prioritization for multitask learning. In: Proc. ECCV 2018. pp. 270–287
34. Guo, Y., Zhang, Z.: LPSE: Lightweight password-strength estimation for password meters. Comput. Secur. **73**, 507–518 (2018)
35. Haan, K.: America's password habits: 46% report having their password stolen over the last year (June 2024), https://www.forbes.com/advisor/business/software/american-password-habits/
36. He, H., Ma, Y.: Imbalanced learning: foundations, algorithms, and applications (2013)
37. Houshmand, S., Aggarwal, S.: Building better passwords using probabilistic techniques. In: Proc. ACSAC 2012. pp. 109–118
38. Islam, M., Bohuk, M.S., Chung, P., Ristenpart, T., Chatterjee, R.: Araña: Discovering and characterizing password guessing attacks in practice. In: Proc. USENIX SEC 2023. pp. 1019–1036
39. Kang, H., Seifer, G., Lee, D., Ryu, J.: Do your best and get enough rest for continual learning. In: Proc. CVPR 2025. pp. 10077–10086
40. Kelleher, J.D., Namee, B.M., D'Arcy, A.: Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. MIT Press (2015)
41. Kudo, T.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226 (2018)
42. Li, Z., Wang, D.: Targeted password guessing using k-nearest neighbors. In: Proc. NDSS 2026
43. Li, Z., Hiratani, N.: Optimal task order for continual learning of multiple tasks. In: Proc. ICML 2025. pp. 34578–34603

44. media, S.: Credential stuffing attacks exacerbate. https://www.scmagazine.com/brief/identity-and-access/credential-stuffing-attacks-exacerbate (Sept 2022)
45. Melicher, W., Ur, B., Segreti, S.M., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F.: Fast, lean, and accurate: Modeling password guessability using neural networks. In: Proc. USENIX SEC 2016. pp. 175–191
46. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
47. Mohri, M., Rostamizadeh, A.: Rademacher complexity bounds for non-iid processes. In: Proc. NeurIPS 2008. vol. 21
48. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of machine learning. MIT press (2018)
49. Monaikul, N., Castellucci, G., Filice, S., Rokhlenko, O.: Continual learning for named entity recognition. In: Proc. AAAI. vol. 35, pp. 13570–13577
50. Nisenoff, A., Golla, M., Wei, M., Hainline, J., Szymanek, H., Braun, A., Hildebrandt, A., Christensen, B., Langenberg, D., Ur, B.: A two-decade retrospective analysis of a university's vulnerability to attacks exploiting reused passwords. In: Proc. USENIX SEC 2023. pp. 5127–5144
51. Pal, B., Daniel, T., Chatterjee, R., Ristenpart, T.: Beyond credential stuffing: Password similarity models using neural networks. In: Proc. IEEE S&P 2019
52. Pal, B., Islam, M., Bohuk, M.S., Sullivan, N., Valenta, L., Whalen, T., Wood, C., Ristenpart, T., Chatterjee, R.: Might I Get Pwned: A second generation compromised credential checking service. In: Proc. USENIX SEC 2022. pp. 1831–1848
53. Pasquini, D., Ateniese, G., Bernaschi, M.: Interpretable probabilistic password strength meters via deep learning. In: Proc. ESORICS 2020. pp. 502—522
54. Qiao, F., Mahdavi, M.: Learn more, but bother less: parameter efficient continual learning. Proc. NeurIPS 2024 **37**, 97476–97498
55. Radauskas, G.: Hackers stole $300,000 from DraftKings users in credential stuffing attack (Nov 2022), https://cybernews.com/news/hackers-draftkings-credential-stuffing-attack/
56. Reichl, D.: KeePass, version 1.26/2.23: Password Quality Estimation (July 2015), http://keepass.info/help/kb/pw_quality_est.html
57. Saeed, N.: Top insights from our 2022 state of secure identity report. https://auth0.com/blog/top-insights-from-our-2022-state-of-secure-identity-report/ (Sept 2022)
58. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Proc. ACL 2017. pp. 1073–1083
59. Shi, Y., Yuan, L., Chen, Y., Feng, J.: Continual learning via bit-level information preserving. In: Proc. CVPR 2021. pp. 16674–16683
60. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proc. NeurIPS 2014 (2014)
61. Thomas, D.R., Pastrana, S., Hutchings, A., Clayton, R., Beresford, A.R.: Ethical issues in research using datasets of illicit origin. In: Proc. IMC 2017. pp. 445–462
62. Toulas, B.: Have I Been Pwned warns of DatPiff data breach impacting millions (Jan 2022), https://www.bleepingcomputer.com/news/security/have-i-been-pwned-warns-of-datpiff-data-breach-impacting-millions/
63. Ur, B., Alfieri, F., Aung, M., Bauer, L., Christin, N., Colnago, J., Cranor, L.F., Dixon, H., Emami Naeini, P., Habib, H., et al.: Design and evaluation of a data-driven password meter. In: Proc. ACM CHI 2017. pp. 3775–3786
64. Ur, B., Kelley, P.G., Komanduri, S., et al.: How does your password measure up? The effect of strength meters on password creation. In: Proc. USENIX SEC 2012

65. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Proc. NeurIPS 2015
66. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. **12**(11), 2776–2791 (2017)
67. Wang, D., He, D., Cheng, H., Wang, P.: fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars. In: Proc. IEEE/IFIP DSN 2016
68. Wang, D., Shan, X., Dong, Q., Shen, Y., Jia, C.: No single silver bullet: Measuring the accuracy of password strength meters. In: Proc. USENIX SEC 2023
69. Wang, D., Wang, P., He, D., Tian, Y.: Birthday, name and bifacial-security: understanding passwords of chinese web users. In: Proc. USENIX SEC 2019
70. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: An underestimated threat. In: Proc. ACM CCS 2016. pp. 1242–1254
71. Wang, D., Zou, Y., Xiao, Y.A., Ma, S., Chen, X.: Pass2Edit: A multi-step generative model for guessing edited passwords. In: Proc. USENIX SEC 2023
72. Wang, D., Zou, Y., Zhang, Z., Xiu, K.: Password guessing using random forest. In: Proc. USENIX SEC 2023. pp. 965–982
73. Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: Theory, method and application. IEEE Trans. Pattern Anal. Machine Intell. **46**(8), 5362–5383 (2024)
74. Wang, M., Michel, N., Xiao, L., Yamasaki, T.: Improving plasticity in online continual learning via collaborative learning. In: Proc. CVPR 2024. pp. 23460–23469
75. Wang, W., Hu, Y., Chen, Q., Zhang, Y.: Task difficulty aware parameter allocation & regularization for lifelong learning. In: Proc. CVPR 2023. pp. 7776–7785
76. Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proc. IEEE S&P 2009. pp. 391–405
77. Wheeler, D.L.: zxcvbn: Low-budget password strength estimation. In: Proc. USENIX SEC 2016. pp. 157–173
78. Winder, D.: 184,162,718 Passwords And Logins Leaked — Apple, Facebook, Snapchat (May 2025), https://www.forbes.com/sites/daveywinder/2025/05/23/184162718-passwords-and-logins-leaked---apple-facebook-snapchat/
79. Xiu, K., Wang, D.: PointerGuess: Targeted password guessing model using pointer mechanism. In: Proc. USENIX SEC 2024. pp. 5555–5572
80. Xu, M., Yu, J., Zhang, X., Wang, C., Zhang, S., Wu, H., Han, W.: Improving real-world password guessing attacks via bi-directional Transformers. In: Proc. USENIX SEC 2023. pp. 1001–1018
81. Yang, T., Wang, D.: Rankguess: Password guessing using adversarial ranking. In: Proc. IEEE S&P 2025. pp. 682–700
82. Zhou, C., Jacobs, T., Gadhikar, A., Burkholz, R.: Pay attention to small weights. In: Proc. NeurIPS 2025
83. Zimmermann, V.: From the quest to replace passwords towards supporting secure and usable password creation. Ph.D. thesis (2021)
84. Zou, Y., An, M., Wang, D.: Password guessing using large language models. In: Proc. USENIX SEC 2025. pp. 7799–7818

## A Missing proofs

We present the proofs of Theorems 1 and 2 in Sec. 4.1. First, we present Lemma 1 stating the generalization upper bound of the Rademacher complexity.

**Lemma 1.** *Let the loss function be bounded with* $|\ell_t(h_t(x), y)| \leq b$ *and satisfy the* $\rho_1$*-Lipschitz condition, and the hypothesis class* $\mathcal{H}_t$ *has its empirical Rademacher complexity as* $\hat{\mathfrak{R}}(\mathcal{H}_t) = \mathbb{E}\big[\sup_{h_t \in \mathcal{H}_t} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h_t(x_i)\big]$ *(where* $\sigma_i$ *is uniformly sampled from* $\{-1, 1\}$*). For the generalization risk in Definition 1,*

$$\mathcal{R}_{gen}(h_t) \leq 2\rho_1 \hat{\mathfrak{R}}(\mathcal{H}_t) + 3b\sqrt{\frac{\ln(2/\eta)}{n}} \tag{9}$$

*holds with the probability* $\geq 1-\eta$ *for all* $h_t \in \mathcal{H}_t$*. The right-hand side is regarded as the upper bound of the generalization risk, and is denoted as* $\mathcal{G}_{t_A \to t_B}$ *if* $t_A$ *and* $t_B$ *are used as initial and subsequent tasks in continual learning.*

*Proof.* We set $\tilde{h}_t(x) = h_t(x)/b$, and obtain the following relationship

$$\mathcal{R}_{gen}(h_t) = \mathbb{E}_{\mathcal{D}_t}\left[\ell_t(\tilde{h}_t(x), y)\right] - \frac{1}{n}\sum_{i=1}^{n} \ell_t(\tilde{h}_t(x_i), y_i) \leq 2\hat{\mathfrak{R}}(\ell(\mathcal{H}_t/b)) + 3\sqrt{\frac{\ln(2/\eta)}{n}} \tag{10}$$

via the upper bound of the Rademacher complexity. By multiplying $b$ and using the Lipschitz condition, we can obtain Eq. 9. This concludes the proof.

We now show the proof of our main theorems.

**Theorem 1.** *Let* $t_A \to t_B$ *denote the case where the initial and subsequent tasks are* $t_A$ *and* $t_B$*, respectively. For the generalization risk, their upper bounds have* $\mathcal{G}_{t_A \to t_B} \leq \mathcal{G}_{t_B \to t_A}$ *if there are* $\mathcal{H}_{t_A} \supseteq \mathcal{H}_{t_B}$ *and* $n_{t_A} \leq n_{t_B}$ *for their hypothesis classes and sample sizes, respectively. For the forgetting risk, if the gradient and Hessian matrix of the initial-task loss function against the model parameter* $\boldsymbol{\theta}$ *satisfy* $\|\nabla \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\| = 0$ *and* $\|\nabla^2 \mathcal{L}_s(\boldsymbol{\theta}) - \nabla^2 \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\| \leq \rho_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{pt}}\|$ *for some* $\rho_2$ *in the vicinity of the pre-trained model* $\boldsymbol{\theta}_{\mathrm{pt}}$*, then there exists*

$$\mathcal{R}_{forget}(t_A, t_B) \leq \frac{1}{2}\Big(\lambda_{\max} \nabla^2 \mathcal{L}_s(\boldsymbol{\theta}_{\mathrm{pt}})\Big) \|\boldsymbol{\delta}\|^2 \tag{3}$$

*where* $\lambda_{\max}$ *is the maximum eigenvalue of the Hessian matrix and* $\boldsymbol{\delta} = \boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{pt}}$ *is the model parameter update near* $\boldsymbol{\theta}_{\mathrm{pt}}$*.*

*Proof.* For the generalization risk, since $\mathcal{H}_B \subseteq \mathcal{H}_A$ indicates $\hat{\mathfrak{R}}(\mathcal{H}_B) \leq \hat{\mathfrak{R}}(\mathcal{H}_A)$ for the empirical Rademacher complexity, so we have the following by Lemma 1,

$$\mathcal{G}_{t_A \to t_B} - \mathcal{G}_{t_B \to t_A} = 2\rho_1(\hat{\mathfrak{R}}(\mathcal{H}_B) - \hat{\mathfrak{R}}(\mathcal{H}_A)) + 3b(\ln(2/\eta))^{\frac{1}{2}}(n_B^{-\frac{1}{2}} - n_A^{-\frac{1}{2}}) \leq 0, \tag{11}$$

i.e., there exists $\mathcal{G}_{t_A \to t_B} \leq \mathcal{G}_{t_B \to t_A}$ for task selections.

For the forgetting risk, we use the Taylor expansion and the mean value theorem on the loss function of the initial task. Thus, for some $a \in (0, 1)$, we have

$$\begin{aligned}\mathcal{R}_{forget}(t_A,t_B) &= \mathcal{L}_s(\theta_{\rm pt}+\boldsymbol{\delta})-\mathcal{L}_s(\theta_{\rm pt}) = \nabla\mathcal{L}_s(\theta_{\rm pt})^\top\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^\top\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt}+a\boldsymbol{\delta})\boldsymbol{\delta}\\ &= \frac{1}{2}\boldsymbol{\delta}^\top\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt})\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^\top\big(\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt}+a\boldsymbol{\delta})-\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt})\big)\boldsymbol{\delta}\\ &\le \frac{1}{2}\Big((\boldsymbol{\delta}U^\top)\Lambda(U^\top\boldsymbol{\delta})+\rho_2\|\boldsymbol{\delta}\|^3\Big)\le\frac{1}{2}\Big(\lambda_{\max}\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt})+\rho_2\|\boldsymbol{\delta}\|\Big)\|\boldsymbol{\delta}\|^2\\ &\approx \frac{1}{2}\Big(\lambda_{\max}\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt})\Big)\|\boldsymbol{\delta}\|^2\end{aligned} \tag{12}$$

where the third line comes from the orthogonal decomposition of the Hessian matrix $\nabla^2\mathcal{L}_s(\boldsymbol{\theta}_{\rm pt})=U\Lambda U^T$ with $\Lambda=\mathrm{diag}(\lambda_1,\ldots,\lambda_d)$ This concludes the proof.

**Theorem 2.** *Let the pre-trained model $\boldsymbol{\theta}_{\rm pt}$ for the initial task $t_A$ be partitioned into frozen (F) and unfrozen (UF) subsets with the update $\boldsymbol{\delta}$ being $\boldsymbol{\delta}^{\rm full}=(\boldsymbol{\delta}_F^{\rm full},\boldsymbol{\delta}_{\rm UF}^{\rm full})$ and $\boldsymbol{\delta}^{\rm frz}=(\mathbf{0},\boldsymbol{\delta}_{\rm UF}^{\rm frz})$ for the subsequent task $t_B$. When using full fine-tuning and partial fine-tuning, the upper bound of the generalization risks has $\mathcal{G}^{\rm frz}<\mathcal{G}^{\rm full}$ if their hypothesis classes have $\mathcal{H}_{\rm frz}\subseteq\mathcal{H}_{\rm full}$, and the difference in the forgetting risks of the initial task has the following*

$$\begin{aligned}&\mathcal{R}^{\rm full}_{forget}(t_A,t_B)-\mathcal{R}^{\rm frz}_{forget}(t_A,t_B)\\ &=\frac{1}{2}(\boldsymbol{\delta}_F^{\rm full})^\top H_{F,F}\boldsymbol{\delta}_F^{\rm full}+(\boldsymbol{\delta}_F^{\rm full})^\top H_{F,\rm UF}\boldsymbol{\delta}_{\rm UF}^{\rm full}+\frac{1}{2}\Big[(\boldsymbol{\delta}_{\rm UF}^{\rm full})^\top H_{\rm UF,UF}\boldsymbol{\delta}_{\rm UF}^{\rm full}-(\boldsymbol{\delta}_{\rm UF}^{\rm frz})^\top H_{\rm UF,UF}\boldsymbol{\delta}_{\rm UF}^{\rm frz}\Big],\end{aligned} \tag{4}$$

*with the Hessian matrix $H_S=\nabla^2\mathcal{L}_S(\boldsymbol{\theta}_{\rm pt})=\begin{bmatrix}H_{F,F} & H_{F,\rm UF}\\ H_{\rm UF,F} & H_{\rm UF,UF}\end{bmatrix}$.*

*Proof.* For the generalization risk, the proof is similar to that of Theorem 1, we omit it here. For the forgetting risk, we apply Theorem 1 to updates of both full and partial (i.e., without and with parameter frozen) fine-tunings, and obtain

$$\mathcal{R}^{\rm full}_{forget}(t_A,t_B)=\frac{1}{2}(\delta^{\rm full})^T H_S\delta^{\rm full},\qquad \mathcal{R}^{\rm frz}_{forget}(t_A,t_B)=\frac{1}{2}(\delta^{\rm frz})^\top H_S\delta^{\rm frz}. \tag{13}$$

By expanding the full quadratic form, we can further obtain

$$(\boldsymbol{\delta}^{\rm full})^\top H_S\boldsymbol{\delta}^{\rm full}=(\boldsymbol{\delta}_F^{\rm full})^\top H_{F,F}\boldsymbol{\delta}_F^{\rm full}+2(\boldsymbol{\delta}_F^{\rm full})^\top H_{F,\rm UF}\boldsymbol{\delta}_{\rm UF}^{\rm full}+(\boldsymbol{\delta}_{\rm UF}^{\rm full})^\top H_{\rm UF,UF}\delta_{\rm UF}^{\rm full}. \tag{14}$$

The frozen case retains only the last term. Subtracting these expressions and dividing by 2 leads to the stated gap in Eq. 4. This concludes the proof.

## B Detailed setups for original reuse-based PSMs and their continual learning under VersaPSE

Vec-PPSM [51] estimates password strength against credential tweaking using similarity metrics and word embeddings. Specifically, we use the authors' source code[3] to train its FastText model [15] on users' passwords from the corresponding training sets, i.e., Tianya for Chinese scenarios and Rockyou for English datasets.

[3] See https://github.com/Bijeeta/credtweak

PR-PSM [79] employs various heuristics for strength evaluation. We train the SentencePiece model [41] it relies on using Tianya (Chinese) and Rockyou (English) datasets. As the original work does not specify how to compute the importance distribution for simulated guess ranks, we use a uniform distribution over the $10^3$ guesses. This simplification affects fewer than 0.5% of passwords in our experiments, so the resulting deviations are negligible.

BERT-PSM [80] employs conditional password guessing (CPG) and targeted password guessing (TPG) models on the client side, and deploys adaptive rule-based password guessing (ARPG) models on the server-side. We mainly focus on CPG and TPG since sending plain-text passwords to the server is not privacy-preserving. Additionally, as the CPG model only provides a visual hint for the strength impacted by each character of the password, it does not affect the guess number output by BERT-PSM. Hence, we use the output guess number of the TPG model as the output strength of BERT-PSM. We follow the settings of the original work via the source code provided by the authors[4], and further implement a torch version of PassBERT for better performance and compatibility.

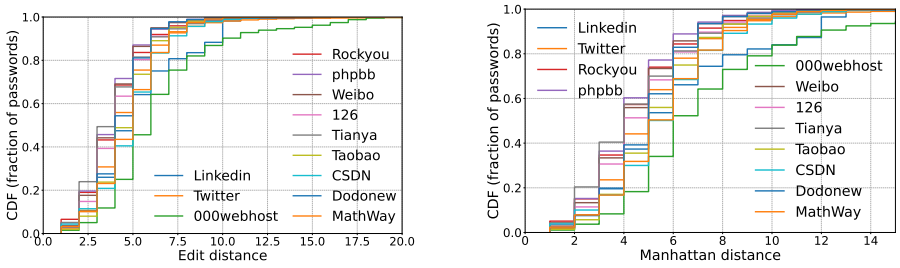
KNN-PSM [42] utilizes Retrieval Augmented Generation (RAG) techniques to enhance the precision of the Transformer model in credential tweaking attacks. We use the open-sourced code provided by the authors in our experiments[5].

We next explicate the layer freezing strategy we adopt for each credential tweaking model, under the principles we propose in VersaPSE.

- **KNNGuess [42].** This model employs a Transformer encoder-decoder architecture. The encoder takes in the given password and learns its semantic feature as well as the distribution behind it, while the decoder learns how users modify passwords and outputs new passwords character-by-character. Hence, we freeze the decoder during the fine-tune phase, and further train the encoder on reuse behavior based on popular passwords to learn broader semantic features.
- **PointerGuess [79].** The backbone of PointerGuess is Pointer Network, where an encoder-decoder structure and a pointer module are utilized [65]. Like KNNGuess [42], the encoder learns semantics of the given passwords, the pointer module captures how users retain characters from old passwords, and the decoder models how users select new characters. During the fine-tuning, we freeze both the pointer module and the decoder to preserve the knowledge of character retention and selection learned from complex local reuse behaviors during continual learning.
- **Pass2Edit [71].** As a multi-step decision-making model, Pass2Edit uses three layers of GRU for choosing the edit operations at every time step. Since lower layers learn high-level features of passwords, and higher layers learn low-level and more detailed features, we freeze the first layer of GRU to keep more complex and detailed semantic features that influence edit operation choices, as well as the fully-connected layers that directly output edit operations based on the outputs of the GRU.

[4] See https://github.com/snow0011/PassBertStrengthMeter

[5] See https://github.com/KNNGuess/KNNGuess-code




Edit Distance distribution Manhattan Distance distribution

**Fig. 10:** Similarity between users' passwords and *popular passwords*. For the Levenshtein and Manhattan distances, a smaller distance indicates higher similarity. We can see that a large proportion of users' passwords are similar to popular passwords.

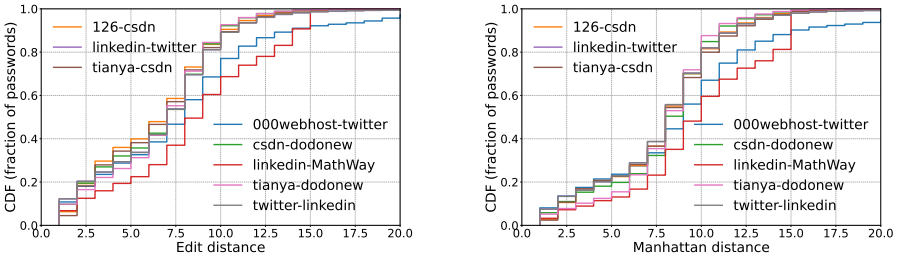
- **PassBERT [80]** uses bi-directional Transformers to first learn representations of password semantics, then classifies edit operations for each character for the given password. Since the classification decision is directly influenced by the fully-connected layers and the pooler, we freeze these parts along with the lowest two layers of the transformer decoder.
- **Pass2Path [51]** treats password reuse behavior as applying a sequence of edit operations on the given password, and employs a Seq2Seq [60] encoder-decoder architecture as the backbone. The encoder learns the semantics of the input password, and the decoder directly outputs the sequence of edit operations. Hence, we freeze the decoder during continual learning on the subsequent task (global reuse behavior).

## C Exploring reuse behavior with other similarity metrics

Besides 2-gram cosine similarity, LCS, and overlap score, we also use Levenshtein distance (Edit distance) and Manhattan distance to explore reuse behavior. The results are shown in Figs. 10 and 11. When the similarity threshold is 5 for both distances (i.e., considered similar when the distance is below 5), 43.2%∼54.7% of users' passwords are similar to the popular passwords, suggesting that users also reuse popular passwords upon password creation. The average Levenshtein (resp. Manhattan) distance between a user's password and a most similar popular password is 4.56 (resp. 5.24), while that between a user's password and her sister password is 6.74 (resp. 8.05), and these results are statistically significant ($p$-value$\ll$0.01 in Kolmogorov-Smirnov and Mann-Whitney U tests). This demonstrates that users' behavior of reusing popular passwords is less complex than reusing sister passwords.

## D Evaluating reuse-based PSMs with other metrics

We presented the experiment results using Balanced Accuracy in credential tweaking attack scenarios in Sec. 5.3. Here, we also present the results using Area-Under-Curve and F1-score in Tables 6 and 7, respectively, and these results are consistent with the results presented using Balanced Accuracy.




Edit Distance distribution Manhattan Distance distribution

**Fig. 11:** Similarities of the sister password pair chosen by the same user across different services. The similarity between sister passwords is significantly lower than that between users' passwords and popular passwords, highlighting that modifying users' sister passwords is a more complex user behavior.

**Table 6:** AUC for credential tweaking attack scenarios

| Scenario | VersaPSE (KNNG.)* | VersaPSE (Pass2Edit)* | VersaPSE (PointerG.)* | VersaPSE (PassBERT)* | VersaPSE (Pass2Path)* | KNN-PSM | PR-PSM | BERT-PSM | Vec-PPSM |
|---|---|---|---|---|---|---|---|---|---|
| #1 | 0.976 (+13.5%∼ +80.8%) | 0.967 (+12.4%∼ +79.0%) | 0.933 (+8.4%∼ +72.7%) | 0.850 (-1.2%∼ +57.3%) | 0.960 (+11.6%∼ +77.7%) | 0.695 [0.860] | 0.718 | 0.540 [0.703] | 0.755 [0.833] |
| #2 | 0.950 (+5.7%∼ +56.5%) | 0.952 (+5.9%∼ +56.8%) | 0.927 (+3.1%∼ +52.7%) | 0.886 (-1.4%∼ +46.0%) | 0.939 (+4.4%∼ +54.6%) | 0.607 [0.894] | 0.848 | 0.636 [0.899] | 0.652 [0.784] |
| #3 | 0.987 (+4.6%∼ +82.0%) | 0.982 (+4.2%∼ +81.2%) | 0.973 (+3.2%∼ +79.5%) | 0.934 (-0.9%∼ +72.4%) | 0.962 (+2.0%∼ +77.5%) | 0.542 [0.784] | 0.943 | 0.786 [0.907] | 0.738 [0.814] |
| #4 | 0.947 (+6.8%∼ +67.6%) | 0.943 (+6.3%∼ +66.8%) | 0.920 (+3.7%∼ +62.9%) | 0.879 (-0.9%∼ +55.5%) | 0.931 (+4.9%∼ +64.8%) | 0.601 [0.887] | 0.799 | 0.565 [0.866] | 0.644 [0.749] |
| #5 | 0.986 (+6.3%∼ +71.7%) | 0.976 (+5.3%∼ +70.1%) | 0.973 (+5.0%∼ +69.5%) | 0.947 (+2.2%∼ +65.0%) | 0.964 (+4.0%∼ +68.0%) | 0.574 [0.802] | 0.908 | 0.815 [0.927] | 0.753 [0.845] |
| #6 | 0.969 (+6.1%∼ +47.3%) | 0.951 (+4.2%∼ +44.6%) | 0.940 (+3.0%∼ +42.9%) | 0.915 (+0.2%∼ +39.1%) | 0.954 (+4.5%∼ +44.9%) | 0.803 [0.913] | 0.690 | 0.658 [0.800] | 0.761 [0.826] |
| #7 | 0.972 (+7.5%∼ +53.8%) | 0.962 (+6.4%∼ +52.2%) | 0.934 (+3.3%∼ +47.8%) | 0.932 (+3.1%∼ +47.5%) | 0.959 (+6.1%∼ +51.7%) | 0.831 [0.904] | 0.673 | 0.632 [0.759] | 0.788 [0.829] |
| #8 | 0.950 (+8.0%∼ +34.8%) | 0.930 (+5.7%∼ +32.0%) | 0.934 (+6.1%∼ +32.5%) | 0.890 (+1.1%∼ +26.2%) | 0.923 (+4.9%∼ +31.0%) | 0.729 [0.880] | 0.817 | 0.750 [0.866] | 0.705 [0.827] |
| Average | 0.967 (+11.8%∼ +43.7%) | 0.958 (+10.7%∼ +42.3%) | 0.942 (+8.9%∼ +39.9%) | 0.904 (+4.5%∼ +34.3%) | 0.949 (+9.7%∼ +41.0%) | 0.673 [0.865] | 0.799 | 0.673 [0.841] | 0.724 [0.813] |

**Table 7:** F1-Score for credential tweaking attack scenarios

| Scenario | VersaPSE (KNNG.)* | VersaPSE (Pass2Edit)* | VersaPSE (PointerG.)* | VersaPSE (PassBERT)* | VersaPSE (Pass2Path)* | KNN-PSM | PR-PSM | BERT-PSM | Vec-PPSM |
|---|---|---|---|---|---|---|---|---|---|
| #1 | 0.820 (+8.2%∼ +454.0%) | 0.830 (+9.5%∼ +460.9%) | 0.726 (-4.3%∼ +390.3%) | 0.700 (-7.7%∼ +372.8%) | 0.807 (+6.5%∼ +445.6%) | 0.559 [0.758] | 0.604 | 0.148 [0.505] | 0.388 [0.454] |
| #2 | 0.841 (+2.8%∼ +131.7%) | 0.858 (+4.9%∼ +136.3%) | 0.820 (+0.2%∼ +125.9%) | 0.763 (-6.7%∼ +110.2%) | 0.840 (+2.7%∼ +131.5%) | 0.363 [0.784] | 0.818 | 0.429 [0.805] | 0.569 [0.654] |
| #3 | 0.953 (+1.4%∼ +502.9%) | 0.949 (+1.1%∼ +500.7%) | 0.935 (-0.5%∼ +491.5%) | 0.873 (-7.0%∼ +452.5%) | 0.906 (-3.5%∼ +473.5%) | 0.158 [0.731] | 0.939 | 0.727 [0.840] | 0.703 [0.740] |
| #4 | 0.858 (+12.2%∼ +269.8%) | 0.851 (+11.2%∼ +266.7%) | 0.826 (+8.0%∼ +256.2%) | 0.752 (-1.7%∼ +224.0%) | 0.826 (+7.9%∼ +255.9%) | 0.340 [0.765] | 0.747 | 0.232 [0.732] | 0.567 [0.640] |
| #5 | 0.925 (+4.3%∼ +254.4%) | 0.919 (+3.6%∼ +251.9%) | 0.894 (+0.8%∼ +242.6%) | 0.835 (-5.9%∼ +219.9%) | 0.868 (-2.2%∼ +232.5%) | 0.261 [0.565] | 0.887 | 0.771 [0.859] | 0.519 [0.617] |
| #6 | 0.864 (+1.7%∼ +79.7%) | 0.821 (-3.4%∼ +70.7%) | 0.795 (-6.4%∼ +65.4%) | 0.743 (-12.6%∼ +54.5%) | 0.819 (-3.6%∼ +70.3%) | 0.751 [0.850] | 0.551 | 0.481 [0.658] | 0.489 [0.580] |
| #7 | 0.866 (+1.1%∼ +106.6%) | 0.827 (-3.4%∼ +97.3%) | 0.769 (-10.2%∼ +83.5%) | 0.770 (-10.0%∼ +83.8%) | 0.822 (-4.0%∼ +96.2%) | 0.790 [0.856] | 0.513 | 0.419 [0.561] | 0.518 [0.572] |
| #8 | 0.835 (+5.2%∼ +78.0%) | 0.823 (+3.6%∼ +75.5%) | 0.806 (+1.5%∼ +71.8%) | 0.744 (-6.2%∼ +58.7%) | 0.800 (+0.7%∼ +70.5%) | 0.625 [0.789] | 0.757 | 0.665 [0.794] | 0.469 [0.614] |
| Average | 0.870 (+14.2%∼ +80.9%) | 0.860 (+12.8%∼ +78.7%) | 0.821 (+7.8%∼ +70.8%) | 0.772 (+1.4%∼ +60.6%) | 0.836 (+9.7%∼ +73.8%) | 0.481 [0.762] | 0.727 | 0.484 [0.719] | 0.528 [0.609] |

Results in green denote the respective multipurpose PSM derived using VersaPSE outperform all other prior reuse-based PSMs in the corresponding scenario.

Results in parenthesis, e.g., (+11.8% ∼ +72.2%) are the improvements of our multipurpose PSMs compared with existing reuse-based PSMs. Results in brackets, e.g., [0.688] denote the performance of existing reuse-based PSMs under the assistance of Zxcvbn-PSM.

* VersaPSE denotes an instantiation of VersaPSE, with the model in the brackets being the adopted credential tweaking model.