Manipulating Uncertainty Ranking with Poisoning Attacks for Adversarial Concept Drift

***Abstract*—Deployed machine learning models often suffer performance degradation over time due to distribution shifts between training and testing data, a phenomenon known as concept drift. To address this, researchers have developed Concept Drift Adaptation with Active Learning (CDA-AL) to improve model robustness. CDA-AL adapts the model to evolving data distributions by labeling and retraining on concept drift samples selected from the testing data. However, the impact of poisoning attacks on CDA-AL's sample selection remains underexplored.**

In this paper, we investigate the robustness of CDA-AL, focusing on its vulnerability to data poisoning attacks. We design a novel attack that consistently causes the victim model to misclassify attacker-specified concept drift samples, which we refer to as attack targets. Specifically, this attack manipulates the victim model's uncertainty ranking on the testing data by injecting crafted poisoned samples, which wastes the active learning labeling budget on less informative data and leads to misclassification of attack targets. Additionally, the proposed attack uses naturally occurring concept drift samples to generate poisoned samples, reducing the cost in data construction.

We evaluate the attack on four concept drift datasets, achieving an average success rate exceeding 80%. We further conduct an in-depth analysis of the longest-spanning real-world malware concept drift dataset in our evaluation. We find that even under constrained attacker capabilities, the attack maintains a high success rate. Furthermore, we analyze the impact of different sample selection strategies and labeling budgets on its attack effectiveness. Our analysis also shows that four existing countermeasures fail to stop the attack. To address this, we propose an adaptive sample filtering method based on intra-cluster distance.

I. Introduction

Machine learning models are widely applied across various practical domains [1]–[4]. However, as the gap between the testing and training data distributions widens over time, the model's performance on the testing data declines [5], [6]. This phenomenon is known as concept drift [7]–[9], which poses severe threats in critical domains, such as industrial risk analysis and malware detection [10], [11]. To mitigate these risks, addressing the challenge of concept drift has become a hot topic of current research [12]–[15].

A widely used approach for Concept Drift Adaptation is based on Active Learning (CDA-AL) [16]–[18]. Active Learning is a machine learning paradigm where the model selectively queries labels for samples expected to be the most informative [19], [20]. CDA-AL applies this principle to detect potential concept drift samples in the testing data using a sample selection strategy. One common method selects samples with the highest classifier uncertainty [16], treating them as informative. These high-uncertainty samples, referred to as concept drift samples, are manually labeled and incorporated into the training data for model retraining. Obviously, providing ground-truth labels for high-uncertainty samples incurs a high cost, which is quantified as the labeling budget [21], [22].

Despite these advantages, previous studies have revealed that the concept drift adaptation process may be vulnerable to poisoning attacks. For instance, prior work by Korycki et al. [23] demonstrated that injecting poisoned samples into the victim model's training data can compromise the performance. However, their approach relies on the strong assumption that attackers can arbitrarily inject poisoned samples into the victim model's training data [24]. In real-world scenarios, attackers typically have access only to the testing data and lack direct control over the training data. Furthermore, in CDA-AL methods [16], [25], [26], only high-uncertainty samples are manually reviewed and labeled before being added to the training set. As a result, the risk of poisoning attacks in CDA-AL, where adversaries cannot freely inject training data, remains underexplored.

In this paper, we propose a novel Poisoning Attack against Concept Drift Adaptation with active learning (PACDA). The core idea is to hinder the victim model from learning informative knowledge about the attack target through its concept drift adaptation process. In PACDA, the attacker manipulates the uncertainty ranking of unlabeled testing data, misallocating the victim model's limited labeling budget to samples with lower informational value. PACDA is a type of resource-exhaustion attack that poses a significant risk, as the labeling budget and annotation time will be depleted even if domain experts later detect the poisoning. Moreover, existing poisoning attacks on concept drift adaptation degrade overall model performance and lack stealth [23]. In contrast, PACDA selectively consumes part of the labeling budget for its target, preserving the victim model's overall performance and enhancing attack stealth.

The effectiveness of PACDA is evaluated across four concept drift datasets, namely APIGraph [27], BODMAS [28], MNIST [29], and SPAM [30], achieving an average success rate of over 80%. Among these, APIGraph, which features the longest time span and largest number of attack targets, is the primary dataset for our experimental validation. For instance,

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.[23|24]xxxx
www.ndss-symposium.org

over a 72-month period, we conducted attacks on more than 1,000 individual targets and achieved an average success rate of 88%. To better understand the effectiveness of PACDA under different conditions, we further analyze key factors influencing attack effectiveness, including sample selection strategies of active learning, the victim model's labeling capacity, and the attacker's ability to identify valuable attack targets. Notably, our attack remains effective across all settings, and we provide an in-depth analysis of the relationship between influencing factors and attack performance.

We further evaluate four defense techniques, including Activation Clustering [31], Data-Free Pruning [32], and other widely adopted techniques for defending against poisoning attacks. Experimental results indicate that these methods struggle to effectively distinguish between poisoned and clean samples, leading to the failure of defenses against PACDA. These findings highlight the urgent need for new defense mechanisms against PACDA. To address this, we propose a poisoned sample filtering approach based on intra-cluster distance to mitigate the effects of our PACDA, reducing the attack success rate by nearly 40%. The contributions of this paper are as follows:

- We present PACDA, the first poisoning attack targeting concept drift adaptation with active learning. PACDA causes the misallocation of labeling budget to uninformative data by manipulating the uncertainty ranking of testing data, leading to persistent misclassification of attack targets during the adaptation process.
- We introduce a strategy for generating high-uncertainty poisoned samples using naturally occurring concept drift examples as seeds. By applying problem-space and feature-space perturbations, our method generates enough poisoned samples to exhaust the limited labeling budget.
- We extensively evaluate the PACDA across diverse datasets, model architectures, and sample selection strategies, achieving consistently high attack success rates. To demonstrate our commitment to open-source, we have made an anonymized version of the code available at https://anonymous.4open.science/r/PACDA-Attack-B730
- We identify critical limitations in existing defense mechanisms against PACDA. We present a poisoned sample filtering method based on cluster-adaptive distance to mitigate this attack.

II. Preliminaries

In this section, we introduce the concept drift adaptation process based on active learning and a taxonmy of existing data poisoning attacks.

A. Concept Drift Adaptation

Concept drift adaptation based on active learning is a paradigm that enhances model performance by identifying high-uncertainty samples from the testing data [16], [25], [26]. For consistency, we refer to the concept drift adaptation model as the victim model throughout this paper. The entire concept drift process is composed of multiple concept drift cycles. Let $f_{\boldsymbol{\theta}_n}$ denote the victim model trained on the training data $\boldsymbol{D}_{tr}^n$ in concept drift cycle n. $\boldsymbol{\theta}_n$ denotes the retrained model parameters at the end of concept drift cycle n, while $\boldsymbol{D}_{tr}^n$ refers to the updated training dataset used by the victim model during that cycle. The unlabeled testing dataset $\boldsymbol{D}_{te}^n$ comprises all samples collected by the victim model throughout concept drift cycle n. For concept drift cycle n, the concept drift adaptation process generally consists of three steps:

Step I: Victim Model $f_{\boldsymbol{\theta}_{n-1}}$ performs inference on the input $\mathbf{x}_i \in \boldsymbol{D}_{te}^n$ and obtain the classification confidence vector $\mathbf{c}_i = f_{\boldsymbol{\theta}_{n-1}}(\mathbf{x}_i)$. The different dimensions of $\mathbf{c}_i$ indicate the model's confidence that the input $\mathbf{x}_i$ belongs to a specific sample label. The label with the highest confidence is selected as the predicted label $\overline{y}_i$ on the input $\mathbf{x}_i$.

Step II: The uncertainty score $\mathbf{u}_i$ of $\mathbf{x}_i$ is measured. For example, uncertainty can be measured as one minus the maximum softmax output of the network [16]: $\mathbf{u}_i = 1 - \max(\mathbf{c}_i, 1 - \mathbf{c}_i)$. We use $uncer()$ to denote uncertainty measures in the following discussion.

Step III: High-uncertainty samples are selected from the testing data $\boldsymbol{D}_{te}^n$ as concept drift samples $\boldsymbol{D}_{dr}^n$. The size of $\boldsymbol{D}_{dr}^n$ is determined by the manual labeling capacity, referred to as the labeling budget β. Then, the concept drift data $\boldsymbol{D}_{dr}^n$ is manually labeled to get the ground truth label and added to the existing training data $\boldsymbol{D}_{tr}^{n-1}$ to form an updated training data $\boldsymbol{D}_{tr}^n = \boldsymbol{D}_{tr}^{n-1} \cup \boldsymbol{D}_{dr}^n$. The victim model $f_{\boldsymbol{\theta}_{n-1}}$ is then retrained on the updated training data $\boldsymbol{D}_{tr}^n$ to yield an updated model $f_{\boldsymbol{\theta}_n}$, as described in Equation 1.

$$\boldsymbol{\theta}_n = \arg\min_{\boldsymbol{\theta}_{n-1}} \sum_{\mathbf{x}_i \in \boldsymbol{D}_{tr}^n} \mathcal{L}\left(f_{\boldsymbol{\theta}_{n-1}}(\mathbf{x}_i), y_i\right) \tag{1}$$

B. Data Poisoning Attacks

In data poisoning attacks, the attacker constructs poisoned data to degrade the performance of the victim model. The most common strategy is to inject poisoned samples into the victim model's training data. As noted by Tian et al. [33] and Wang et al. [24], such attacks are generally categorized as targeted or untargeted. Untargeted poisoning attacks aim to hinder the convergence of the victim model and eventually lead to denial-of-service [34]–[36]. The challenge with untargeted attacks is that they aim to degrade the performance across all data [37]. Therefore, the effect of poisoned samples must outweigh that of the clean training data. In contrast, targeted poisoning attacks aim to manipulate the victim model into making incorrect predictions on specific inputs [38]–[40]. The inputs for which the model's predictions are compromised are called attack targets. To launch a targeted poisoning attack, an adversary needs to inject malicious knowledge into the training data while keeping other knowledge unaffected [33].

III. Threat Model, Design Challenegs and Overview of PACDA

In this section, we present the threat model, the design rationale behind the attack, and an overview of PACDA.

A. Threat Model

Similar to previous research [41], [42], attackers' goal is to ensure that the victim model consistently misclassifies the attack target throughout the CDA-AL process. The attacker will also attempt to keep the victim model's overall performance stable, thereby maintaining the stealth of the attack. In addition, it is important to note that in the PACDA, the attack target has a defined source class but does not enforce misclassification into a specific target class. By avoiding a fixed target class, this class-agnostic misclassification strategy increases the difficulty of detection, as the victim model is more likely to interpret the misclassifications as effects of natural concept drift or inherent uncertainty rather than deliberate poisoning attack.

We assume attackers cannot access the victim model's internal parameters or influence its training process, including manual labeling [43]. Attackers can only submit samples to the victim model for querying to obtain outputs such as sample uncertainty scores and predicted labels. Additionally, attackers are presumed to have access to public data and the resources required to train surrogate models [44], [45]. Consistent with previous research on active learning attacks [46], we assume the attacker typically has knowledge of the victim model's labeling budget settings.

B. Design Challenges

PACDA faces three key challenges. (1) The sample selection strategy deployed in CDA-AL restricts the arbitrary injection of poisoned samples into the training data. Therefore, attackers must ensure that the victim model selects poisoned samples for labeling. (2) The sample labeling process in active learning is based on manual analysis, meaning the labels of all poisoned samples in the training data must remain correct. The attacker cannot degrade the victim model's concept drift adaptation performance by tampering with the sample labels [42], [47]. (3) Existing targeted poisoning attacks are typically evaluated for effectiveness on models with fixed parameters. For example, backdoor attacks are often evaluated on pre-trained models [38]–[40]. In contrast, PACDA must ensure its attack remains effective as the victim model continuously updates its parameters. PACDA tackles the first two challenges. The final challenge is addressed through continuous attacks on the concept drift adaptation process.

C. Design Rationale

The key issue is ensuring that the poisoned samples have high uncertainty. Prior research has demonstrated that uncertainty quantification techniques are vulnerable to manipulation by adversaries [48]. Specifically, Ledda et al. employed perturbation search techniques to find the minimal perturbations capable of altering a sample's uncertainty [48]. However, their method focuses on reducing uncertainty, whereas CDA-AL selects samples with high uncertainty [16]. Inspired by Yang et al. and Chen et al., who represent sample uncertainty by measuring the similarity between testing and training data [16], [49], we find that newly appearing concept drift samples always inherently exhibit high uncertainty. This is primarily because of their low similarity to the existing training data. Therefore, our attack exploits the inherent high uncertainty of concept drift samples to craft poisoned samples that are likely to be selected for labeling, thereby exhausting the labeling budget and preventing the victim model from learning the attack target.

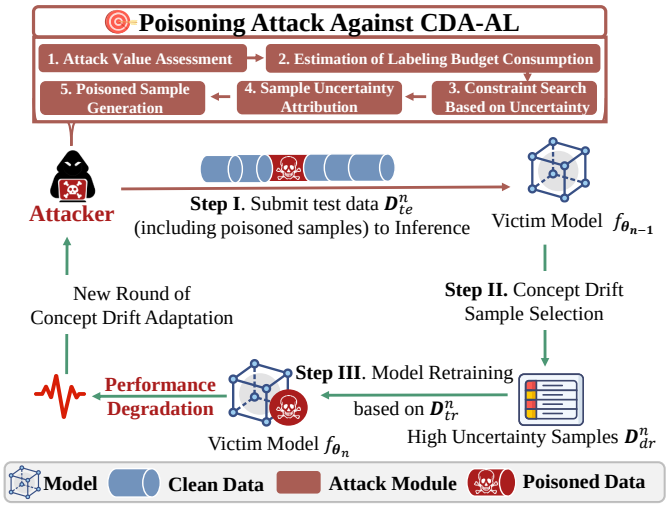

Fig. 1: Poisoning Attack against CDA-AL

D. Overview of PACDA

Based on the design rationale, we propose the PACDA framework, illustrated in Figure 1. First, we perform a value assessment of the attack target to avoid wasting limited attack resources on targets that do not hold attack value. Next, we estimate the required budget consumption for labeling based on the selected attack target. By controlling the budget consumed, the attacker determines which knowledge the victim model can or cannot acquire. Subsequently, we define the search constraints and identify high-uncertainty samples from the testing data as poisoning seeds to consume the victim model's labeling budget. We then perform uncertainty-based feature attribution on selected testing samples to prepare for the subsequent generation of poisoned samples. As the number of such seeds is often insufficient to fully consume the labeling budget, we further adopt a poisoned sample generation strategy. PACDA provides two generation strategies. Strategy II enhances Strategy I but incurs higher cost. The attacker selects between them based on knowledge of the attack target (e.g., uncertainty ranking). Ultimately, PACDA exhausts the victim model's labeling budget, preventing effective learning of the attack target.

IV. Detailed Design of PACDA

We demonstrate the design of the PACDA method by using concept drift cycle n as an example, executing the same attack procedure in every concept drift cycle. In concept drift cycle n, the complete testing dataset collected by the victim model is denoted as $\boldsymbol{D}_{te}^n \in \mathbb{R}^{N \times d}$ (with N samples in d dimensions).

The attack target sample $\mathbf{x}_{tar}$ is a member of the testing data $\boldsymbol{D}_{te}^{n}$. The victim model is denoted as $f_{\boldsymbol{\theta}_{n-1}}$, and its labeling budget is represented by β. Notably, at the start of concept drift cycle n, the victim model is derived from the model updated at the end of concept drift cycle $n-1$. Therefore, the model parameters at this stage are denoted as $\boldsymbol{\theta}_{n-1}$. The victim model performs uncertainty quantification on the collected testing data and ranks the samples accordingly. The top β samples with the highest uncertainty ranking are selected for manual analysis to obtain the concept drift data $\boldsymbol{D}_{dr}^{n} \in \mathbb{R}^{\beta \times d}$ (with β samples in d dimensions).

A. Attack Value Assessment

The PACDA is essentially a resource-exhaustion attack that targets the labeling budget of the victim model. However, the attacker also faces similar risks of resource exhaustion. For example, resources may be wasted if the attacker chooses a attack target already in the training data. A failed attack prevents the attacker from obtaining any reward and incurs substantial attack costs. Consequently, the attacker must carefully assess whether the attack target is worth attacking. This decision is based on the accuracy of the victim model's prediction of the attack target. Samples not present in the training data are more likely to be misclassified, making this a strong indicator of attack value. The attacker can leverage the victim model to make this determination. If the predicted label $\overline{y}_i$ of $\mathbf{x}_{tar}$ from the victim model $f_{\boldsymbol{\theta}_{n-1}}$ differs from the ground-truth label y_i, the attack target is likely to be of high attack value.

B. Estimation of Labeling Budget Consumption

Once the attack target is determined to be valuable, the PACDA begins by estimating the amount of the labeling budget that needs to be exhausted from the victim model. The attacker makes decisions based on the uncertainty ranking of the attack target. If the uncertainty ranking r_{tar} of the attack target is higher than the labeling budget boundary ($r_{tar} < r_{\beta}$), the attack target will be selected as a concept drift sample. It is important to note that a higher uncertainty ranking corresponds to a smaller value of r_{tar}. Then, the attack target will be learned during the retraining process of the victim model. The gap between the uncertainty ranking of the attack target and the labeling budget boundary represents the primary objective of the attacker's labeling budget consumption, denoted as C_n. It also reflects the minimum requirement for the number of poisoned samples. When $r_{tar} > r_{\beta}$, it indicates that the attack target $\mathbf{x}_{tar}$ will not be selected by the victim model $f_{\boldsymbol{\theta}_{n-1}}$. Nonetheless, the attacker will still generate some poisoned samples, even if the attack target is not selected. This mitigates potential mismatches between the attacker's and the victim's testing data, which could otherwise cause the attacker to overestimate the target's uncertainty ranking. Therefore, to ensure reliability, the consumption of the labeling budget is set to a fixed value $\lambda\,(\lambda \ll \boldsymbol{D}_{te}^{n})$, determined by the attacker's computational resources. Specifically, the amount of budget consumption C_n for labeling is as follows:

$$C_n = \begin{cases} (r_{\beta} - r_{tar}) + 1, \; r_{tar} < r_{\beta} \\ \lambda, \; r_{tar} > r_{\beta} \end{cases} \tag{2}$$

C. Constraint Search based on Uncertainty Ranking

The attacker then searches the testing data $\boldsymbol{D}_{te}^{n}$ for naturally occurring samples that can be used to exhaust the victim model's labeling budget C_n. These selected samples are referred to as poisoning seed samples. The poisoning seed data, denoted as $\boldsymbol{D}_{seed}^{n} \in \mathbb{R}^{K \times d}$ (with K samples in d dimensions), consists of samples that satisfy the uncertainty criterion defined in Equation 3. We denote each row of the testing data as a sample vector $\mathbf{x}_i$, where $\mathbf{x}_i = \boldsymbol{D}_{te,\,i*}^{n}, \forall i \in \{0, \ldots, N-1\}$. The uncertainty of these poisoned seed samples must exceed the attack target's uncertainty, as shown below.

$$uncer(f_{\boldsymbol{\theta}_{n-1}}(\mathbf{x}_i)) > uncer(f_{\boldsymbol{\theta}_{n-1}}(\mathbf{x}_{tar})) \tag{3}$$

Furthermore, it is crucial to ensure that the poisoned samples exhaust the victim model's labeling budget without enhancing its concept drift adaptation capability. For instance, incorporating novel malware samples into the poisoning seed data may improve the victim model's ability to detect malicious behavior, thereby undermining the attacker's objective of causing misclassification of the attack target. Therefore, the attacker must remove malware samples from the attack seed data to avoid reducing the attack success rate on the attack target. In addition, similarity-based filtering can be employed to prevent the inclusion of samples similar to the attack target in the poisoned samples.

$$(y_j \neq y_{tar}) \wedge [sim(\mathbf{x}_j, \mathbf{x}_{tar}) < \tau] \tag{4}$$

Each row of the poisoning seed data $\boldsymbol{D}_{seed}^{n}$ is denoted as a sample vector $\mathbf{x}_j$, where $\mathbf{x}_j = \boldsymbol{D}_{\text{seed}}^{n}[j,:], \forall j \in \{0, \ldots, K-1\}$. The constraint condition applied to each sample vector $\mathbf{x}_j$ is defined in Equation 4. Samples that do not satisfy this condition are removed from the poisoning seed data $\boldsymbol{D}_{seed}^{n}$. Finally, the refined poisoning seed data is sorted in descending order of uncertainty, with the sample at index 0 having the highest uncertainty.

D. Sample Uncertainty Attribution

Next, the attacker constructs an uncertainty interpreter based on Shapley Additive Explanations (SHAP) to facilitate the subsequent poisoned sample generation strategy. The fundamental concepts of SHAP are provided in Appendix A-B. We utilize a standardized permutation-based SHAP method to interpret sample uncertainty, approximating shapley values by iterating through permutations of input features. This model-agnostic explainer ensures local accuracy (additivity) by iterating completely through an entire permutation of the features in both forward and reverse directions (antithetic sampling). Since the objective of the attack is to prevent the victim model from learning the attack target, the generated poisoned samples must suppress the model's ability to adapt

to concept drift. To achieve this, samples outside the labeling budget are selected as targets for feature space perturbation. Their lower importance relative to in-budget samples helps avoid inadvertently strengthening the model's adaptation capabilities. The attacker applies uncertainty feature attribution on the testing data outside the labeling budget, denoted as $\boldsymbol{D}^n_{shap}$ ($\boldsymbol{D}^n_{shap} = \boldsymbol{D}^n_{te} \setminus \boldsymbol{D}^n_{dr}$). These samples are then sorted in descending order based on their uncertainty scores.

$$\boldsymbol{V}_{shap} = SHAP(\boldsymbol{D}^{n-1}_{tr}, UncerSort(\boldsymbol{D}^n_{shap})) \quad (5)$$

$\boldsymbol{V}_{shap}$ represents the matrix of uncertainty feature attributions for data $\boldsymbol{D}^n_{shap}$. $\mathbf{v}_i$ denotes the feature attribution vector of a specific sample, while $v_{i,j}$ indicates the impact of a particular feature on the uncertainty of the given sample.

E. Poisoned Sample Generation

The poisoned seed samples $\boldsymbol{D}^n_{seed}$ is incorporated into the victim model's training data due to its high uncertainty. However, the seed samples obtained via the constrained search strategy may not fully satisfy the required labeling budget consumption C_n of the victim model. Consequently, attack targets $\mathbf{x}_{tar}$ will still be selected and labeled manually, leading to a failure of the PACDA. To ensure the effectiveness of the PACDA, the attacker must generate an additional batch of poisoned samples such that the total number of poisoned samples is greater than or equal to the labeling budget consumption C_n.

1) Selection of Poisoned Sample Generation Strategy: We propose two poisoned sample generation strategies: Strategy I, which is based on problem-space perturbation, and Strategy II, which leverages Shapley additive explanations. Both strategies aim to generate poisoned samples that consume the labeling budget, thereby leading to misclassification of the attack target. However, the two strategies are suitable for different types of attack targets. Strategy I prevents the victim model from learning specific knowledge about the attack target by constructing poisoned samples without introducing incorrect concept drift information. It focuses solely on preventing the victim model from learning the attack target while minimizing its impact on the model's ability to adapt to concept drift. In contrast, Strategy II prevents the victim model from acquiring knowledge about the attack target and introduces misleading information about concept drift into the training process. Although this reduces the model's ability to adapt to true concept drift in the testing data, it does not cause a significant drop in overall performance, as indicated by the F1 score in Table IV.

In terms of strategy selection, Strategy I is better suited for attack targets that are inherently challenging to learn, typically those exhibiting a high degree of concept drift. Conversely, when the attack target is relatively easy to learn due to weak concept drift, strategy II is employed to enhance the effectiveness of the attack. Finally, we ensure that, regardless of the degree of concept drift exhibited by the attack target, the victim model fails to learn the attack target due to its inability to efficiently allocate the limited labeling budget, ultimately leading to the misclassification of the attack target.

2) Problem-Space Perturbation (Strategy I): Specifically, Strategy I is particularly effective for rapidly and cost-efficiently adjusting the uncertainty rankings of samples within the labeling budget. The problem space perturbation strategy refers to modifying the samples in the poisoning attack seed data $\boldsymbol{D}^n_{seed}$ to generate new poisoned samples without altering the features of the samples. We denote each row of the attack seed data $\boldsymbol{D}^n_{seed}$ as a sample vector $\mathbf{x}_k$, where $\mathbf{x}_k = \boldsymbol{D}^n_{\text{seed}}[k, :]$,$\forall k \in \{0, \ldots, M-1\}$. Due to the need for attack stealth, the samples generated by problem space perturbation must minimize the impact on other concept drift samples. Therefore, we select the ϵ least uncertain samples from the poisoned seed sample set for problem space perturbation. The smaller ϵ is, the lesser the impact on existing concept drift samples. In this paper's subsequent attack effectiveness evaluation, we set ϵ to 5, which accounts for 0.025 of the labeling budget. A perturbation in the problem space is applied to these samples $\mathbf{x}_k$, $k \in \{0, \ldots, \epsilon - 1\}$, resulting in a new data denoted as $\boldsymbol{D}^n_{\alpha}$. α denotes the problem-space perturbation operations, as shown in Table I. This strategy ensures that the newly generated poisoned samples exhibit high uncertainty, thereby maintaining their potential to exhaust the victim model's labeling budget and increase the uncertainty ranking of the attack target r_{tar} beyond the labeling budget (i.e., $r_{tar} > r_\beta$).

TABLE I: Problem-Space Perturbation Operations

Dataset	**Perturbation Operations** (α)
APIGraph	Rename method names to meaningless identifiers
APIGraph	Modify image and other resource files
BODMAS	Modify the system API function names
BODMAS	Dynamically adjust the size of the DOS STUB space
MNIST	Add Gaussian noise to the image pixels
MNIST	Apply sharpening to enhance the edges of the image
SPAM	Remove non-essential words
SPAM	Insert random symbols or additional spaces

The reason is that machine learning models, during the feature extraction process, often ensure that non-critical perturbations α in the data do not affect the features in order to construct robust representations. Therefore, when the attacker adjusts the non-essential information of a sample, the sample is altered, but its features remain unchanged. Since existing uncertainty quantification methods are based on sample features, altering the problem space does not reduce the sample's uncertainty, ensuring that the labeling budget can be exhausted. For example, in software applications, when key information such as permissions and API calls is extracted as features, elements like icon usage, coding style, and redundant code have no direct relationship with these critical features. Examples of problem space perturbations in different domains can be found in Table I. All perturbation operations preserve the original labels of the poisoned seed samples. The perturbation operation in the problem space is infinite, allowing for generating a sufficient number of

poisoned samples. Thus, by leveraging high-uncertainty seed samples $\boldsymbol{D}^n_{seed}$ and the problem space perturbation strategy, the attacker can generate poisoned samples $\boldsymbol{D}^n_{\alpha}$ to meet the victim model's labeling budget consumption requirements C_n. This attack strategy generates poisoned samples quickly and is low-cost, making it the preferred choice for most attack targets. As long as the poisoned sample seed set is not empty, problem space perturbation can be applied.

3) Feature-Space Perturbation (Strategy II): However, the uncertainty of poisoned samples generated through problem space perturbation is limited by the uncertainty of the seed samples from $\boldsymbol{D}^n_{seed}$. Thus, it is difficult to affect samples with higher uncertainty than the poisoned seed samples. If the portion of the labeling budget with higher uncertainty than the poisoning seed samples contains samples that hinder the misclassification of the attack target (e.g., malware using similar vulnerabilities as the target), the attack success rate may decrease. We introduce a feature space perturbation method (Strategy II) based on the uncertainty attribution matrix $\boldsymbol{V}_{shap}$ to overcome this limitation. This approach removes the upper bound constraint on poisoned sample uncertainty imposed by the poisoned seed samples. It is worth noting that the uncertainty attribution matrix allows samples originally outside the labeling budget to be shifted into the budget through feature-space perturbation. Consequently, the approach eliminates the dependence on high-uncertainty seed samples. It enhances the attack's robustness by preventing the victim model from acquiring accurate knowledge of actual concept drift samples. Feature space perturbation can not only overcome the inherent limitations of problem space perturbation but also be combined with it, making it a viable approach to further enhance the effectiveness of problem space perturbation strategies.

The uncertainty attribution module has obtained the uncertainty attribution matrix $\boldsymbol{V}_{shap}$. We denote each row of the uncertainty attribution matrix as a vector $\mathbf{v}_i$, where $\mathbf{v}_i = \boldsymbol{V}_{\text{shap}}[i,:], \forall i \in \{0,\ldots,N-1\}$. Based on the uncertainty feature attribution vector $\mathbf{v}_i$ for each sample, the attacker identifies the set of feature indices $\boldsymbol{I}_{shap}$ that have the greatest impact on increasing uncertainty. The computation of the feature index vector $\boldsymbol{I}_{shap}$ is shown in Equation 6.

$$\boldsymbol{I}_{shap} = \{argsort(\mathbf{v_i})[0:d-1]\} \tag{6}$$

d denotes the dimension of the feature vector for each sample. The sorting function ranks feature indices based on their influence on sample uncertainty, such that features contributing more to increased uncertainty are assigned smaller index values. Therefore, the attacker can focus on modifying features with lower index values to effectively amplify uncertainty. We selected the top 2% of features based on their indices for modification. Since the uncertainty-based SHAP attribution provides a linear approximation of the model's predictive uncertainty, modifying the key features with the highest attribution values increases the sample's overall uncertainty.

A sample from the APIGraph dataset is taken as an example, and its features are represented as 0-1 vectors. This feature modification corresponds to adjusting the API call patterns in the actual application. Since the poisoned samples are intended to consume the labeling budget, the impact of the API calls on the program's functionality does not need to be considered. Moreover, the modifications only reduce software API calls without introducing additional sensitive behaviors, thereby preserving the original label of the sample. For data in other domains, we replace the values of the important features with the corresponding values from samples with high uncertainty. The perturbed samples generated in this manner are considered as a new set of poisoned attack samples $\boldsymbol{D}^n_{shap}$. Suppose the generated poisoned samples are smaller than the labeling budget consumption requirements C_n. In that case, the attacker can further amplify the set by applying problem space perturbation to the poisoned samples $\boldsymbol{D}^n_{shap}$.

Moreover, the number of poisoned samples only needs to be sufficient to consume the victim model's labeling budget. Consequently, they constitute only a small fraction of the training data. For example, in the real-world APIGraph [27] data, poisoned samples generated in a single attack account for only 1% of the training data. In addition to the stealth provided by the low poisoning ratio, all poisoned samples retain correct labels without any label flipping.

V. Evaluation

A. Experimental Setup

All experiments are conducted on a Windows 11 system with 96GB memory, one Intel® Core™ i7-14700K 3.4GHz CPU, and an NVIDIA GeForce RTX 4080 SUPER (16GB). **Concept Drift Datasets:** The PACDA evaluation is conducted on four datasets: two synthetic datasets (MNIST [29] and Spam-Email [30]) and two real-world datasets (APIGraph [27] and BODMAS [28]). Detailed dataset information is presented in Table II. Android malware datasets span 7 years and naturally exhibit concept drift. In contrast, non-timestamped datasets such as MNIST are typically partitioned into sub-datasets to simulate concept drift artificially. The method for synthesizing concept drift is similar to those used in existing concept drift studies [50]. By clustering the dataset based on sample similarity and selecting a subset of similar samples for initial training, we simulate a realistic scenario where less similar data is gradually introduced in the subsequent testing data. Synthetic dataset creation details are in Appendix A-C1.

TABLE II: Concept Drift Datasets for Attack Evaluation

Dataset	Type	Duration	Size
APIGraph [27]	Android Malware	7 years	320,315
BODMAS [28]	Windows Malware	2 years	149,217
MNIST [29]	Image Classification	5 cycles	70,000
SPAM [30]	Spam Emails	8 cycles	9324

In addition, concept drift datasets require the testing data to be partitioned according to the order of occurrence. Following prior work [16], we adopt a similar setup. For datasets with timestamps, the APIGraph [27] dataset serves as an example:

TABLE III: Android Concept Drift Dataset (APIGraph)

Year	Malware	Benign	Malware Family
Train-2012	3,061	27,472	104
Test-2013 (Cycle 1)	4,854	43,714	172
Test-2014 (Cycle 2)	5,809	52,676	175
Test-2015 (Cycle 3)	5,508	51,944	193
Test-2016 (Cycle 4)	5,324	50,712	199
Test-2017 (Cycle 5)	2,465	24,847	147
Test-2018 (Cycle 6)	3,783	38,146	128
Total	30,804	289,511	1,118

data from 2012 is used for training, while data from 2013 to 2018 is used for testing, as illustrated in Table III. In the subsequent concept drift experiments, the testing data of the APIGraph dataset is released progressively monthly. For datasets without timestamps, the testing data is divided into multiple approximately equal-sized segments presented to the model sequentially. The detailed training and testing splits for the other three datasets are provided in Appendix A-D.

Victim Models: The victim model's configuration consists of two main parts: the model architecture and the sample selection strategy of active learning. For experiments on the Android malware dataset APIGraph [27], we employ both traditional models (e.g., SVM) and deep learning models (e.g., ResNet). The settings of sample selection strategies follow existing research on concept drift adaptation [16], [49], [51], [52]. For the other datasets (MNIST, SPAM, and BODMAS), the victim model is a multilayer perceptron consisting of five hidden layers, one output layer, and LeakyReLU activation functions. To prevent overfitting, the model includes batch normalization and dropout layers. For the sample selection strategy, we use the classic confidence-based method [52].

APIGraph: The model was trained for 50 epochs using the SGD optimizer with a learning rate (LR) of 0.003, batch size of 1024, and hi-dist-xent loss. Learning rate decay (0.05) was applied at epochs 10, 500, and 10. MNIST: The model was trained for 5 epochs with the Adam optimizer (LR = 0.0004), batch size of 64, and triplet-my loss. No learning rate decay was applied. BODMAS and SPAM-Email: Both models used the AdamW optimizer (LR = 0.0004), batch size of 64, and binary cross-entropy (BCE) loss. BODMAS was trained for 50 epochs and SPAM-Email for 5 epochs. No learning rate decay was applied.

Attack Target: The attacker aims to continuously misclassify newly emerging concept drift samples during testing while maintaining the victim model's overall performance. So, attack targets are selected from test samples emerging during the concept drift adaptation process. For example, in the malware dataset APIGraph [27], new malware samples from the testing data are chosen as attack targets. Appendix A-E provides detailed attack target information. In the real world, multiple attack targets may exist at the same concept drift time point, such as multiple new malware samples emerging within the same month as attack targets. Therefore, we classify the evaluation of attack effectiveness into two types: single-target attack and multi-target attack, as shown in Section V-B. This setup aims to understand better the impact of different attack target settings on attack success rates in real-world scenarios.

Metrics: The effectiveness of PACDA is evaluated using the following metrics: 1) F1 Score: The harmonic mean of precision and recall, providing a balanced measure of the model's overall performance on the testing data. 2) Attack Effectiveness: Effectiveness is assessed via the attack success rate (ASR), where success is defined as the victim model misclassifying the attack target. Additionally, since attack targets may already be misclassified during the early stages of concept drift, we adopt a more stringent criterion for measuring attack success. Only when the PACDA prolongs the duration of the attack target's misclassification will the attack be regarded as successful, as defined in Equation 8.

$$Judge(y_{tar},\overline{y}_{tar},n) = \begin{cases} 0, y_{tar} = \overline{y}_{tar}\, at\ \text{n}, \\ 1, y_{tar} \neq \overline{y}_{tar}\, at\ \text{n}. \end{cases} \tag{7}$$

$$ASR(y_{tar},\overline{y}_{tar},n,N) = \frac{\sum_{n=1}^{N} Judge(y_{tar},\overline{y}_{tar},n)}{N} \tag{8}$$

N denotes the PACDA testing cycle length for the attack target $\mathbf{x}_{tar}$. Function $Judge()$ compares the ground truth label y_{tar} of $\mathbf{x}_{tar}$ with the victim model's predicted label $\overline{y}_{tar}$. A mismatch between them during the testing cycle indicates a successful attack. Since all experiments in this study require running multiple testing cycles, the testing phase incurs significant time overhead. Therefore, in subsequent experimental evaluations, the testing cycle extension is set to 1 for all cases except for dedicated attack persistence tests, which utilize a full 100% testing cycle extension. All reported performance metrics are averaged over five attack runs per target. A small standard deviation reflects the consistency and stability of the results.

B. Attack Effectiveness

The effectiveness of PACDA is evaluated on four datasets. Due to its long time span and real-world dataset, the APIGraph dataset [27] contains the most attack targets. So, we use it for both single-target attacks and multi-target attacks. The manually synthesized concept drift datasets (MNIST [29] and SPAM [30]), as well as the BODMAS malware concept drift dataset [28], contain a limited number of attack targets and span a short time period. Therefore, they are utilized to evaluate multi-target attack scenarios.

1) Single-Target Attack: We assess the effectiveness of the PACDA with a simple attack target on the APIGraph [27] dataset. The evaluation of attack effectiveness involves 724 attack targets across more than 100 malware families. Appendix A-E provides detailed information on the attack targets. To efficiently demonstrate the effectiveness of the PACDA, we first apply a low-cost problem-space perturbation strategy to generate poisoned samples. Subsequently, by employing a feature-space perturbation-based poisoned sample generation strategy, we enhance the poisoning process for attack targets where the problem-space perturbation strategy proves ineffective. This setup allows us to illustrate better how perturbations

in the feature space can enhance the effectiveness of problem-space perturbations.

TABLE IV: Concept Drift Datasets for Attack Evaluation

Attack Target	ASR(%)	F1 score	ACC(%)
Mecor (Trojan-Spy)	100%	0.85 (-0.07)	97.49(-1.07)
Mobidash (Adware)	100%	0.90 (-0.02)	98.28 (-0.28)
Svpeng (Banking Trojan)	100%	0.90 (-0.02)	98.26 (-0.30)
Smforw (Trojan-Spy)	100%	0.81 (-0.11)	96.80 (-1.76)
Fobus (Data Stealing)	60%	0.88 (-0.04)	97.84 (-0.72)
Adflex (Adware)	100%	0.90 (-0.02)	98.26 (-0.30)
Vnapstore (Unknown)	100%	0.85 (-0.07)	97.41 (-1.15)
Clicks (Trojan-Spy)	100%	0.90 (-0.02)	98.40 (-0.16)
Mogap (Trojan-Spy)	100%	0.90 (-0.02)	98.24 (-0.32)
Congur (Trojan-Spy)	100%	0.91 (-0.01)	98.40 (-0.16)

The proportion of poisoned samples in the monthly testing data averages less than 6%, demonstrating a high level of attack stealth.

The PACDA demonstrates high effectiveness, achieving an average ASR of 88%. This implies that the PACDA can extend the misclassification duration of most attack targets during the concept drift adaptation process. In Table IV, we present the top 10 malware families of attack targets with the most significant number of samples, along with their attack success rates and the performance metrics of the victim model. In addition to the fact that 90% of the attack target families achieve an ASR of 100%, we focus on whether the impact of the PACDA on the victim model's performance is sufficiently minimal to enhance its attack stealth. We observe that the F1 fluctuations remain within 0.2, while the ACC stays above 95%. The model's average TNR under PACDA reaches 99%, ensuring minimal false alarms. Based on the above data, it can be concluded that the poisoned sample generation strategy based on problem-space perturbation effectively prevents the victim model from learning the attack target while maintaining its overall performance during the concept drift adaptation process. We further measured the problem-space perturbation time for programs of varying sizes. Experiments on several widely-used programs show that the average perturbation time remains under 100 seconds. In addition, the process can be supported by a range of well-established tools, highlighting the cost-effectiveness of problem-space perturbation.

TABLE V: Time overhead of problem-space perturbation

APK	Size (MB)	Time Overhead
JD	97.59	54.95s
Taobao	57.03	78.98s
Little Red Book	162.99	178.68s
Google	315.67	93.32s
Wang VPN	45.51	14.91s
WeChat	264.04	136.76s

Since the victim model is continuously updated during the concept drift adaptation process, it is necessary to test whether our attack's effectiveness persists over time. To ensure a fair comparison across different attack targets, we standardized the testing period. The testing cycle for each attack target is extended to 200% of the original duration, based on the initial misclassification results, as shown in Appendix A-F. For instance, if the attack target is detected as malicious in the fourth month after its first appearance, we conduct an 8-month attack effectiveness test. The attack is considered successful only if the PACDA extends the target's misclassification duration for the whole 8 months. To illustrate attack persistence trends, we selected malware samples per month from January 2013 to December 2018 as attack targets. Over 1,000 targets were tested during the 6-year concept drift adaptation process, achieving an average ASR of 86.7%, as shown in Figure 2. Therefore, it is evident that the PACDA ensures the attack target remains persistently misclassified throughout the long-term concept drift adaptation process of the victim model. This poses a significant threat to the field of concept drift adaptation, as the core objective of concept drift adaptation methods is to minimize the duration of misclassification.

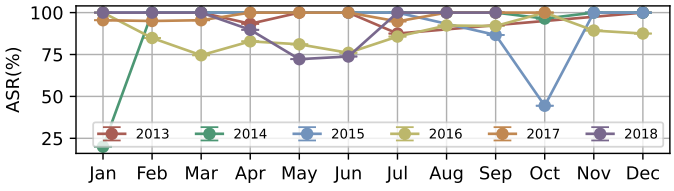

Fig. 2: Attack Persistence over 6 years

The effectiveness of the attack using the problem-space perturbation strategy in poisoned sample generation has already been validated. Next, we evaluate the effectiveness of the feature-space perturbation strategy. We selected 138 families with less than six months of survival for comparative analysis. These samples, characterized by their shorter survival times and weaker concept drift attributes, are more suitable for demonstrating the effect of the feature-space perturbation strategy. We divided the attack targets into five groups based on the duration of the attack test. The results of the attack's effectiveness are presented in Figure 3. After introducing the

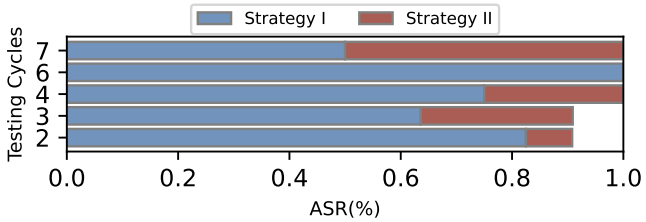

Fig. 3: Feature-space perturbation strategy

SHAP-based feature space perturbation, the attack success rate improved for most targets, reaching an average of 91.3%. This represents a 10.9% increase compared to the previous attack success rate achieved with the problem-space perturbation

strategy. The increase in ASR is due to the attacker's ability to generate poisoned samples with higher uncertainty than those produced using the problem-space perturbation strategy. After applying feature-space perturbation, the average uncertainty ranking of the poisoned samples increased by 66.2%.

Additionally, we further tested the impact on the PACDA when attackers with further limited capabilities as weak Attackers. First, the attacker cannot query the victim model for sample uncertainty scores. In this scenario, the attacker can only obtain the predicted labels from the victim model. Other information, such as sample uncertainty, must be obtained through the surrogate model. So we propose a surrogate model construction method based on knowledge distillation, where the surrogate model's parameters $\boldsymbol{\theta}^*_{n-1}$ approximate those of the victim model $\boldsymbol{\theta}_{n-1}$.

$$\overline{Y} = Q(\boldsymbol{\theta}_{n-1}, \boldsymbol{D}_{te}^{n-1}) \tag{9}$$

The attacker constructs a surrogate model $f_{\boldsymbol{\theta}^*_{n-1}}$ using the input-output query function Q, where the pseudo label set $\overline{Y}$ by querying the victim model $f_{\boldsymbol{\theta}_{n-1}}$ with input data $\boldsymbol{D}_{te}^{n-1}$. The purpose of the surrogate model is to identify the optimal parameters $\boldsymbol{\theta}^*_{n-1}$ such that the prediction error between the surrogate model and the victim model is minimized for all inputs $\mathbf{x}_i \in \boldsymbol{D}_{te}^{n-1}$, as shown in Equation 10.

$$\boldsymbol{\theta}^*_{n-1} = \arg\min_{\boldsymbol{\theta}^*_{n-2}} \sum_{\mathbf{x_i} \in \boldsymbol{D}_{te}^{n-1}} \mathcal{L}\left(f_{\boldsymbol{\theta}^*_{n-2}}(\mathbf{x}_i), \overline{y}_i\right), \overline{y}_i \in \overline{Y} \tag{10}$$

It is important to emphasize that the attacker uses pseudo-labels $\overline{Y}$ generated by the victim model $f_{\boldsymbol{\theta}_{n-1}}$ as training labels for the surrogate model rather than relying on ground truth labels. There are two main reasons for this approach. First, the methods of concept drift adaptation are primarily applied in sensitive domains such as malware detection and industrial security risk analysis, where acquiring ground truth labels is prohibitively expensive. Second, the role of the surrogate model $f_{\boldsymbol{\theta}^*_{n-1}}$ is to approximate the detection capabilities of the victim model $f_{\boldsymbol{\theta}_{n-1}}$, but the victim model does not provide correct labels for the entire testing data $\boldsymbol{D}_{te}^{n-1}$. As a result, the surrogate model, trained with pseudo-labels, performs more similarly to the victim model. When constructing the surrogate model $f_{\boldsymbol{\theta}^*_{n-1}}$, it is important to correctly set the query range for the testing data. In concept drift cycle n, the attacker queries the testing data $\boldsymbol{D}_{te}^{n-1}$, which is obtained from the previous concept drift cycle. This is because the victim model has already completed learning from the previous concept drift cycle. This model is used to quantify the uncertainty of samples in the current concept drift cycle n. Therefore, to ensure that the surrogate model $f_{\boldsymbol{\theta}^*_{n-1}}$ approximates the victim model $f_{\boldsymbol{\theta}_{n-1}}$, the attacker queries the testing data $\boldsymbol{D}_{te}^{n-1}$ and uses the query results to train the surrogate model.

In addition, since computational cost is a primary concern in machine learning scenarios, the attacker's model was weakened to reduce computational overhead. Specifically, we weaken the model by reducing the number of neural network layers in the encoder or classifier. Following state-of-the-art Android malware concept drift adaptation methods [16], which employ an encoder and classifier, we tested four weakened settings based on the victim model, as shown in Figure 4. ENC denotes the encoder, CAL represents the classifier, and W indicates a decrease in the model's capacity, reflected in a reduction of model parameters.

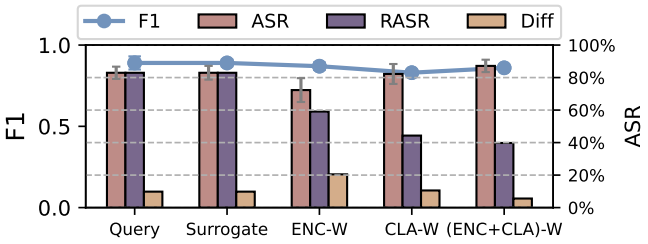

Fig. 4: Attack Effectiveness under Weakened Capabilities

Previous studies have shown that when the attacker's capabilities are limited, the effectiveness of poisoning attacks decreases [24]. However, PACDA mitigates this issue, achieving an average attack success rate of 81.17% with limited attacker capabilities, as shown in Figure 4. We also found that weakening the encoder reduces the attack success rate by nearly 10%, which is more significant than weakening the classifier. This indicates that the encoder part of the surrogate model plays a crucial role in approximating the capabilities of the victim model. Additionally, we found that weakening both the encoder and the classifier in the attacker's surrogate model does not result in the lowest attack success rate. Under this configuration, the attack success rate is even higher than that of the setup where only the classifier is weakened, at 86% compared to 83%. This phenomenon contradicts the expectation that attackers must invest more computational resources to achieve a higher attack success rate. By analyzing the set of attack targets under different configurations, we found that when the surrogate model is weakened, its selection of attack targets is significantly biased. In the synchronized weakening setting, only 55% of the attack targets remain compared to the control group, highlighting the need to assess attack effectiveness under constrained attacker capabilities using ASR and the number of attack targets. To address this issue, we define the Number of Successful Attack Targets (NSAT) and the Relative Attack Success Rate (RASR), which represents the ratio of successfully attacked targets in the weakened attacker's setup to those in the control group, as shown in Equation 11.

$$RASR = NSAT_{weak}/NSAT_{control} \tag{11}$$

As model capability weakens, RASR declines, with the weakest setup showing a 43.18% drop. While a reduced scope of attack targets weakens the attacker's capabilities, the relative attack success rate increases, posing a significant threat to concept drift adaptation by enhancing resource utilization efficiency even under constrained conditions.

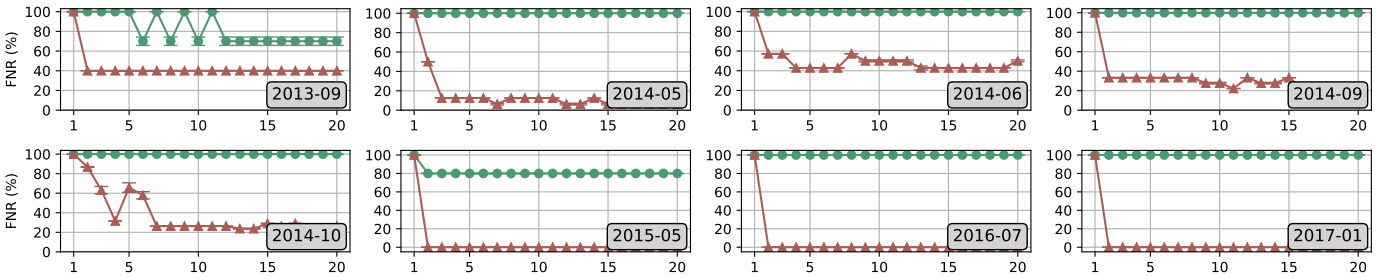

Fig. 5: Attack Effectiveness with Multi-Target Attack (6 year Attack Testing)

2) Multi-Target Attack: Multiple attack targets may emerge within the same concept drift cycle in real-world scenarios. Therefore, PACDA also supports a multi-target attack mode. In the multi-target attack setting, the attacker performs PACDA on multiple attack targets simultaneously. To evaluate the effectiveness of the PACDA in a multi-target setting, we selected 8 months with different attack targets for testing, covering a 6-year evaluation period, as illustrated in Figure 5. For each multi-target setting, we conducted a 20-month attack test. Detailed information on the settings can be found in Appendix A-E. The multi-target attack achieved an average success rate of 97.5% across different testing times, demonstrating its effectiveness in executing the PACDA on multiple targets. During the 80-month attack test, only the attack success rate in May 2015 was not 100%, although it still maintained an effective success rate of 80%. The high attack success rate in a multi-target setting is due to the coverage effect of labeling budget consumption across different attack targets. When multiple attack targets are present, once the labeling budget for the most uncertain attack target is exhausted, the labeling budget for the other attack targets is also concurrently depleted. Consequently, the PACDA incurs a low cost, as a single attack benefits multiple targets simultaneously.

TABLE VI: Multi-Target Attack across Four Ddatasets

Datasets	Targets	F1-Testing	F1-Validation	ASR (%)
APIGraph	734	$0.76_{-0.15}$	0.99	92.92%
MNIST	1,398	$0.78_{-0.12}$	0.99	79.98%
BODMAS	222	$0.92_{-0.06}$	0.92	80.63%
SPAM	168	$0.91_{-0.07}$	0.99	74.40%

In certain special cases, the attacker may need to target all attack targets simultaneously. This represents a special scenario within multi-target attacks. Therefore, we tested the effectiveness of PACDA on all attack targets at different concept drift time points. The average attack success rate reaches 81.98% (as shown in Table VI), with a notably high success rate of 92.92% achieved on the real-world concept drift dataset, APIGraph. Moreover, we observe that the victim models maintain high F1 scores on the initial validation set under attack across different datasets, with an average of 0.99. This demonstrates that our method preserves the victim model's previously learned knowledge, even when targeting all attack targets, thereby highlighting the stealth of the proposed attack. The impact on test performance (F1-Testing) varies across datasets, reflecting how limited access to new samples can hinder the model's adaptation to concept drift. The performance degradation on the BODMAS and SPAM datasets is relatively minor, with average F1-score drops of less than 0.07, indicating a lower intensity of concept drift in these scenarios. In contrast, the APIGraph and MNIST datasets exhibit more substantial performance changes, with average F1-score drops of around 0.14. This suggests stronger concept drift and a higher reliance on the victim model for effective learning from drifted samples. Nevertheless, our attack stealth remains unaffected in both cases, as the testing data is unlabeled, and the victim model cannot promptly detect its performance degradation.

C. Attack Influencing Factors

We have demonstrated the effectiveness of PACDA. To further investigate how real-world conditions affect attack performance, we analyze the factors influencing PACDA using a real-world Android malware dataset (APIGraph [27]) spanning seven years.

1) Impact of Different CDA-AL Strategies: Existing concept drift adaptation strategies in sensitive domains [16], [49], [51] can be broadly categorized into four types. In addition to the uncertainty-based strategies discussed in Section II-A, the remaining approaches fall into three categories. To ensure the generality of our findings, we evaluate the effectiveness of our attack across all these representative adaptation strategies.

- CADE [49] trains an encoder with labeled data to learn compressed input representations. It uses a distance function to identify concept drift samples that deviate from the training data.
$$d_i = ||\mathbf{z}_i - \mathbf{zc}_i||_2 \tag{12}$$
Specifically, $\mathbf{z}_i$ represents the latent space embedding of the sample $\mathbf{x}_i$ obtained from the encoder. A sample is a concept drift instance if its distance d_i to the nearest training sample in the encoder's latent space exceeds a predefined threshold.
- TRANS [51] applies conformal prediction [53] to concept drift adaptation. It calculates a testing sample's nonconformity score, credibility (proportion of calibration samples with higher scores), and confidence (1 minus the opposite label's credibility). Low scores indicate potential drift.

- HCL [16] proposes the latest concept drift adaptation strategy, combining an encoder and classifier. Its training loss $\mathcal{L}(\mathbf{x}_i)$ integrates hierarchical contrast loss $\mathcal{L}_{hc}(\mathbf{x}_i)$ and classification loss $\mathcal{L}_{ce}(\mathbf{x}_i)$.

$$\mathcal{L}(\mathbf{x}_i) = \mathcal{L}_{hc}(\mathbf{x}_i) + \mathcal{L}_{ce}(\mathbf{x}_i) \tag{13}$$

  Samples with higher loss values are more likely to be affected by concept drift, as different samples incur different loss levels during inference.

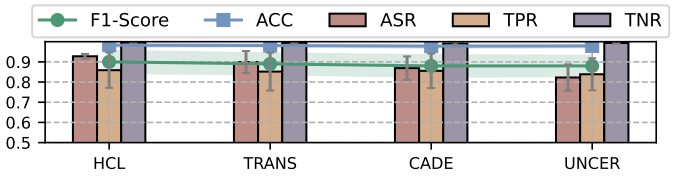

Fig. 6: Different Concept Drift Adaptation Strategies

The evaluation covers traditional uncertainty-based methods and advanced concept drift strategies like contrastive learning. As shown in Figure 6, PACDA achieves over 80% ASR across all four strategies while maintaining F1 scores above 0.88. PACDA attains a 92.77% ASR against the latest adaptation method (HCL).

2) Impact of Labeling Budget: The labeling budget represents the cost of manual labeling and is one of the most valuable resources in CDA-AL. We evaluated the attack effectiveness under six different labeling budget settings. Each labeling budget setting represents its proportion of the testing data, following the settings of existing studies [16]. The first five labeling budget settings assume the attacker knows the victim model's labeling budget. In contrast, the last setting assumes the attacker cannot access the victim model's labeling budget. When the attacker is unaware of the labeling budget settings, poisoned samples are generated based on the maximum computational capacity of the attacker. In this experiment, we set the attacker's capacity four times the potential labeling budget computation capacity. As shown in Figure 7, PACDA remains

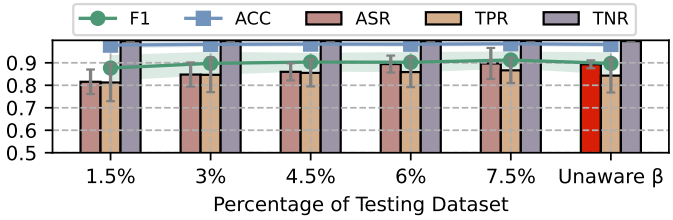

Fig. 7: PACDA Under Different Labeling Budget

effective across budgets, with an average attack success rate of 86.30%. Even without access to the victim model's labeling budget, the attacker can still achieve a high attack success rate by leveraging the maximum available resources, reaching 89.33%. The high attack success rate, even without access to detailed labeling budget information, is due to the significantly lower cost of poisoning attacks than the victim's concept drift adaptation cost, which is primarily driven by labeling expenses. Thus, the attacker can generate as many poisoned samples as possible to consume the victim model's labeling budget. This strategy is inspired by DDoS [54] attacks, where the attacker maximizes attack traffic without knowing the victim's total resources to achieve the attack objective.

3) Impact of Attack Value Assessment: Analyzing the attack value of attack targets provides crucial support for the PACDA. To demonstrate the significance of this step, we conduct ablation experiments where attackers skip the attack value assessment phase and proceed directly to poisoning seed sample selection and poisoned sample generation. This approach implies that attackers target all new malware, leading to a significant increase in attack cost.

TABLE VII: Attack value assessment necessity analysis

CDA-AL Strategies	Ablation-ASR (%)	ASR (%)
HCL (Model Based)	81.73	$92.77_{+11.04}$
CADE (Data Based)	72.43	$86.90_{+14.47}$

Existing concept drift adaptation strategies are primarily divided into data-driven and model-driven approaches. Therefore, in the attack value assessment module, we selected two representative strategies, CADE and HCL, from each category for evaluation. For the labeling budget, we chose a medium-level sample labeling capacity and set the labeling budget to 200. The experimental results reveal that removing the attack value assessment module leads to an average decrease of 15.62% in attack success rates, as shown in Table VII. This highlights the critical role of the attack value assessment module in PACDA.

VI. Potential Defenses

CDA-AL methods lack directly relevant and effective defenses against poisoning attacks. While Lin et al. [55] proposed defenses for active learning poisoning attacks based on static unlabeled and clean datasets. However, these assumptions break down in CDA-AL, where unlabeled data evolves and clean datasets quickly become outdated. Therefore, we evaluated PACDA against four representative poisoning attack defense mechanisms [31], [32], [56], [57].

- **Activation Clustering (AC)** [31] is a data inspection method that assumes poisoned data forms a distinct cluster within the target class, either small or distant from the class center.
- **Data-Free Pruning (DFP)** [32] sanitizes poisoned models by pruning neurons dormant for clean data, assuming poisoned and clean data activate different neurons.
- **Fine-Tuning (FT)** [56] can mitigate targeted poisoning attacks, but it requires extensive labeled training data to avoid overfitting.
- **Fine-Pruning (FP)** [57] mitigates poisoned models by pruning neurons dormant on clean validation data, measuring average neuron activation, and removing the least active ones.

The defense assumptions of FT and FP rely on the victim model having access to a large amount of labeled clean data for fine-tuning or model pruning. However, this assumption is impractical in concept drift adaptation scenarios, where the labeling budget is limited each month, and the training data distribution is continuously evolving. Therefore, the FT and FP methods cannot be executed under the active learning-based concept drift adaptation setting. Under this setting, defenses based on feature separability (AC) and model parameter adjustments (DFP) are applicable, but their defensive effectiveness is limited. As shown in Figure 8, t-SNE visualization reveals that PACDA's use of attack seeds creates intricate entanglement between poisoned and clean samples in the feature space (the APIGraph dataset's test data from June 2013). Consequently, existing defences perform poorly, failing to remove poisoned samples. Thus, the attack success

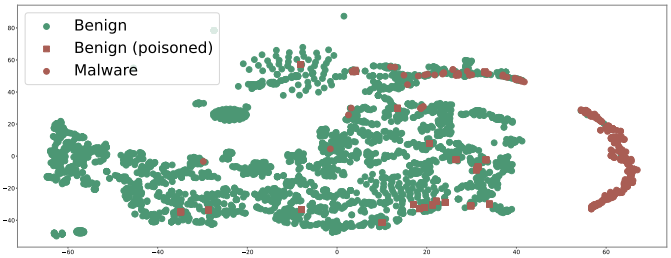

Fig. 8: ICDF Defense Motivation

rate decreases by less than 20% on average. Specifically, the ASR drops by 16.5% under the AC defense and 21.8% under DFP. Given these limitations, we explore alternative defense strategies in countering PACDA.

Intra-Cluster Distance Filtering (ICDF): In the PACDA, the attacker relies on poisoning seed samples to generate poisoned samples, reducing costs and leading to similarity among the poisoned samples. Different poisoned samples may originate from the same seed sample. We measure this similarity using Euclidean distance in the feature space, the basis for our defense method (ICDF). Unlike AC-based defenses, ICDF filters poisoned samples by exploiting the similarities between the poisoned samples themselves. So, we propose an adaptive sample filtering method based on intra-cluster distance. Specifically, using an unsupervised clustering algorithm (e.g., K-means), we partition the concept drift samples into a set of clusters Clu ($Clu = \{c_1, c_2, .., c_k\}$). Each cluster c_k calculates an intra-cluster distance threshold τ (Equation 14) as the mean Euclidean distance $d_{Euc}(\mathbf{x}_i, \mathbf{x}_j)$ between its samples.

$$\tau = \frac{1}{|c_i|(|c_i|-1)} \sum_{i \neq j} d_{Euc}(\mathbf{x}_i, \mathbf{x}_j) \quad i, j \in c_k \tag{14}$$

$|c_i|$ is the number of samples in the cluster c_i. Samples with distances smaller than τ to others in the same cluster are removed. ICDF achieved the best defense effect under single-target attack mode, reducing the success rate of PACDA by 39% across the Top 10 target sets, as shown in Figure 9. While AC and DFP showed some effectiveness, they

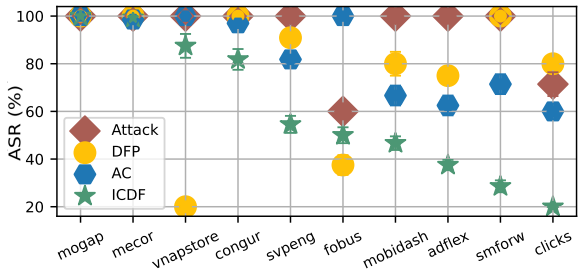

Fig. 9: ICDF Defense (Single-Target PACDA)

were 23.5% and 18.8% less effective than ICDF. We also evaluated our method's defense performance against feature space–based poisoned sample generation strategies on the APIGraph dataset. The attack success rate was reduced by 32.6%, under the same experimental settings as those used in the attack effectiveness evaluation. We also conducted defense tests against multi-target attacks across four different datasets, as shown in Table VIII. The attack success rate decreased by an average of 54.97% across the four datasets, while the F1 score on the mapping dataset was restored to an average of 0.91. Therefore, our ICDF defense method also demonstrates strong effectiveness in defending against multi-target attack scenarios.

TABLE VIII: ICDF Defense (Multi-Target PACDA)

Datasets	Targets	F1-Attack	F1-Defense	ASR-Decrease
APIGraph	734	0.76	0.90 $_{+0.14}$	21.26%
MNIST	1,398	0.78	0.82 $_{+0.04}$	75.47%
BODMAS	222	0.92	0.99 $_{+0.07}$	54.05%
SPAM	168	0.91	0.94 $_{+0.03}$	69.04%

To better demonstrate the effectiveness of the ICDF method, we analyze the variation in defense performance under different parameter settings. We selected the APIGraph dataset, which spans the longest time period, for defense parameter analysis. The number of clusters was the only parameter requiring manual configuration during the defense. We conducted analyses under both single-target and multi-target attack scenarios. We evaluated four cluster settings (6, 10, 20, and 40) under the multi-target attack. The mean F1 score across these four settings was 0.91, with a variance of 0.0025. Given the large number of attack targets in the single-target setting, we selected a subset of representative malware families for detailed evaluation. In the single-target attack, we selected the family 'clicks' (with the best ICDF defense performance) and the family 'mogap' (with the worst ICDF defense performance) from the Top 10 attack targets. For each family, we conducted four experiments with clustering parameter settings of 6, 10, 20, and 40 and observed that the defense performance remained consistent across all settings. This indicates that the defense parameter settings have minimal impact on defense effectiveness, which can effectively reduce the deployment costs of the defense method.

VII. Limitation and Discussion

A. Existence of Attack Seeds

Our attack relies on poisoning seeds to craft effective poisoned samples. Experiments across four datasets show that suitable seeds can be identified for most targets. We also consider extreme cases with no seeds, such as when the attack target is the most uncertain sample in the testing data. In such cases, attackers can delay releasing the attack target and wait for the next concept drift cycle to obtain viable seeds, as long-term misclassification holds more value than immediate impact. Moreover, by leveraging SHAP-based feature space perturbation, we can elevate the uncertainty of less uncertain samples and use them as substitute seeds. Thus, PACDA remains feasible even when seeds are temporarily unavailable.

B. Completeness of Testing Data Collection

To launch our attack, the attacker must collect testing data to compute uncertainty rankings for the attack targets. The completeness of the collected testing data directly affects the accuracy of these rankings. Our design introduces a fixed labeling budget consumption mechanism to address potential uncertainty ranking errors caused by incomplete testing data, as detailed in Section IV-B. Furthermore, in real-world scenarios, attackers can leverage publicly available threat intelligence platforms, such as VirusTotal [58], where new testing data samples are published monthly with unique hash identifiers. This allows attackers to align testing data more effectively, enhancing the efficiency of PACDA.

C. Limitation

Our evaluation of attack effectiveness primarily focuses on the APIGraph [27] dataset, as it spans a long time period and represents a real-world concept drift scenario. However, due to the lack of large-scale, publicly available datasets, our evaluation is limited to synthetic datasets for real-world concept drift scenarios in domains such as image and text. Such scarcity of real-world datasets is also a well-known limitation in existing concept drift research [16], [49], [50].

D. Future Work

Another important future direction is to extend our research to the security of continual learning [59], [60]. As both concept drift adaptation and continual learning are machine learning paradigms designed to address the challenges posed by ever-changing real-world data environments [61], they share similar motivations. However, continual learning typically assumes the emergence of new tasks over time, whereas concept drift assumes a fixed task with shifting data distributions. This key difference motivates our future work, which aims to explore the impact of data poisoning attacks within continual learning frameworks.

VIII. Related Work

A. Poisoning Attacks Against Active Learning

Zhao et al. [62] investigated the security of sampling strategies in active learning. However, their approach assumes that attackers can remove samples from the testing data, preventing the victim model from collecting them. In contrast, attackers typically do not have control over the victim model's data collection process. Miller et al. [63] discussed adversarial threats in active learning, highlighting the risk of manipulation in the sample selection process. However, their attack is indiscriminate and lacks stealth. Vicarte et al. [46] proposed a backdoor attack against active learning, leading to misclassification of specific attack targets. Nevertheless, their method requires embedding triggers into the attack targets, which introduces considerable overhead during the concept drift process. This overhead arises from the need to continuously update the triggers to keep pace with the evolving victim model throughout the adaptation. In contrast, our PACDA requires no modifications to the attack targets.

B. Adversarial Concept Drift

Korycki et al. [23] pointed out that existing drift detectors cannot distinguish between real and adversarial concept drift. Their proposed adversarial concept drift scenario assumes a strong adversary capable of directly manipulating the victim model's training data, which contrasts with our setting, where only selected samples are labeled and used for training in CDA-AL. Apruzzese et al. [64] analyzed the impact of adversarial perturbations occurring simultaneously with real concept drift in the context of intrusion detection. However, their study primarily focuses on the intrusion detection scenario, with limited analysis and validation in other domains. Moreover, their attack methods are limited to the problem space and do not account for the influence of feature space perturbations. In addition, they typically assume that the victim model does not utilize data filtering mechanisms such as active learning.

IX. Conclusion

In this paper, we propose a novel poisoning attack against concept drift adaptation with active learning. Our attack manipulates the uncertainty ranking of testing data by injecting poisoned samples, causing a misallocation of the labeling budget and hindering effective learning of attack targets. To reduce poisoned sample construction costs, we design a generation method that builds on naturally occurring concept drift samples. Our approach also improves the stability of poisoned sample generation by combining problem space perturbations with uncertainty-based feature attribution. Extensive experiments show that CDA-AL is highly vulnerable to our attack, especially in sensitive domains like malware detection. We further analyze existing defenses, expose their limitations, and introduce a novel filtering method tailored to the concept drift adaptation process.

Ethical Considerations

The primary purpose of our research is to evaluate the security of CDA-AL, as related methods have received attention from researchers. Even though the intent is strict about evaluating the weaknesses of CDA-AL, potential ethical concerns are associated with our research. For example, attackers can leverage our methods to improve malware. Therefore, following previous research precedents [65]–[67], we will restrict code sharing to verified academic researchers.

References

[1] S. Liu, X. Li, Z. Mao, P. Liu, and Y. Huang, "Model-driven deep neural network for enhanced aoa estimation using 5g gnb," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 214–221.

[2] Y.-Y. Chang, W.-Y. Wang, and W.-C. Peng, "Sega: Preference-aware self-contrastive learning with prompts for anomalous user detection on twitter," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 30–37.

[3] X. Zeng, X. Hao, H. Tang, Z. Tang, S. Jiao, D. Lu, and J. Peng, "Designing biological sequences without prior knowledge using evolutionary reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 383–391.

[4] C. Cui and C. Jia, "Propagation tree is not deep: Adaptive graph contrastive learning approach for rumor detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 73–81.

[5] N. Malekghaini, E. Akbari, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "Deep learning for encrypted traffic classification in the face of data drift: An empirical study," *Computer Networks*, vol. 225, p. 109648, 2023.

[6] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[8] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[9] D. M. V. Sato, S. C. De Freitas, J. P. Barddal, and E. E. Scalabrin, "A survey on concept drift in process mining," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–38, 2021.

[10] D. W. Fernando and N. Komninos, "Fesad ransomware detection framework with machine learning using adaption to concept drift," *Computers & Security*, vol. 137, p. 103629, 2024.

[11] S. Yang, X. Zheng, J. Li, J. Xu, X. Wang, and E. C. Ngai, "Recda: Concept drift adaptation with representation enhancement for network intrusion detection," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 3818–3828.

[12] M. Karimian and H. Beigy, "Concept drift handling: A domain adaptation perspective," *Expert Systems with Applications*, vol. 224, p. 119946, 2023.

[13] H. Yu, W. Liu, J. Lu, Y. Wen, X. Luo, and G. Zhang, "Detecting group concept drift from multiple data streams," *Pattern Recognition*, vol. 134, p. 109113, 2023.

[14] L. Yang, D. M. Manias, and A. Shami, "Pwpae: An ensemble framework for concept drift adaptation in iot data streams," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 01–06.

[15] E. Yu, J. Lu, B. Zhang, and G. Zhang, "Online boosting adaptive learning under concept drift for multistream classification," in *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, M. J. Wooldridge, J. G. Dy, and S. Natarajan, Eds. AAAI Press, 2024, pp. 16 522–16 530. [Online]. Available: https://doi.org/10.1609/aaai.v38i15.29590

[16] Y. Chen, Z. Ding, and D. Wagner, "Continuous learning for android malware detection," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 1127–1144. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/chen-yizheng

[17] W. Liu, H. Zhang, Z. Ding, Q. Liu, and C. Zhu, "A comprehensive active learning method for multiclass imbalanced data streams with concept drift," *Knowledge-Based Systems*, vol. 215, p. 106778, 2021.

[18] B. Krawczyk, B. Pfahringer, and M. Woźniak, "Combining active learning with concept drift detection for data stream mining," in *2018 IEEE international conference on big data (big data)*. IEEE, 2018, pp. 2239–2244.

[19] B. Settles, "Active learning literature survey," 2009.

[20] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," *ACM computing surveys (CSUR)*, vol. 54, no. 9, pp. 1–40, 2021.

[21] S. Yuan, X. Sun, H. Kim, S. Yu, and C. Tomasi, "Optical flow training under limited label budget via active learning," in *European conference on computer vision*. Springer, 2022, pp. 410–427.

[22] G. Hacohen and D. Weinshall, "How to select which active learning strategy is best suited for your specific problem and budget," *Advances in Neural Information Processing Systems*, vol. 36, pp. 13 395–13 407, 2023.

[23] Ł. Korycki and B. Krawczyk, "Adversarial concept drift detection under poisoning attacks for robust data stream mining," *Machine Learning*, vol. 112, no. 10, pp. 4013–4048, 2023.

[24] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Comput. Surv.*, vol. 55, no. 7, dec 2022. [Online]. Available: https://doi.org/10.1145/3538707

[25] C. H. Park and Y. Kang, "An active learning method for data streams with concept drift," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 746–752.

[26] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.

[27] X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang, "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 757–770.

[28] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "Bodmas: An open dataset for learning based temporal analysis of pe malware," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 78–84.

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[30] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowledge and Information Systems*, vol. 22, pp. 371–391, 2010.

[31] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[32] C. Li, R. Pang, B. Cao, Z. Xi, J. Chen, S. Ji, and T. Wang, "On the difficulty of defending contrastive learning against backdoor attacks," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2901–2918.

[33] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Comput. Surv.*, vol. 55, no. 8, dec 2022. [Online]. Available: https://doi.org/10.1145/3551636

[34] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proceedings of the Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, C.-N. Hsu and W. S. Lee, Eds., vol. 20. South Garden Hotels and Resorts, Taoyuan, Taiwain: PMLR, 14–15 Nov 2011, pp. 97–112. [Online]. Available: https://proceedings.mlr.press/v20/biggio11.html

[35] Y. Yu, Q. Liu, L. Wu, R. Yu, S. L. Yu, and Z. Zhang, "Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4854–4863.

[36] S. Wang and G. Zuccon, "An analysis of untargeted poisoning attack and defense methods for federated online learning to rank systems," in *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, 2023, pp. 215–224.

[37] J. Knauer, P. Rieger, H. Fereidooni, and A.-R. Sadeghi, "Phantom: Untargeted poisoning attacks on semi-supervised learning," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 615–629.

[38] J. Chen, Y. Gao, G. Liu, A. M. Abdelmoniem, and C. Wang, "Manipulating pre-trained encoder for targeted poisoning attacks in contrastive learning," *IEEE Transactions on Information Forensics and Security*, 2024.

[39] G. Severi, J. Meyer, S. Coull, and A. Oprea, "{Explanation-Guided} backdoor poisoning attacks against malware classifiers," in *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 1487–1504.

[40] L. Yang, Z. Chen, J. Cortellazzi, F. Pendlebury, K. Tu, F. Pierazzi, L. Cavallaro, and G. Wang, "Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 719–736.

[41]

[42] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[43] K. Mahmood, R. Mahmood, E. Rathbun, and M. van Dijk, "Back in black: A comparative evaluation of recent state-of-the-art black-box attacks," *IEEE Access*, vol. 10, pp. 998–1019, 2022.

[44] Y. Qin, Y. Xiong, J. Yi, and C.-J. Hsieh, "Training meta-surrogate model for transferable adversarial attack," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, pp. 9516–9524, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26139

[45] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[46] J. R. S. Vicarte, G. Wang, and C. W. Fletcher, "{Double-Cross} attacks: Subverting active learning systems," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1593–1610.

[47] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *International conference on machine learning*. PMLR, 2019, pp. 7614–7623.

[48] E. Ledda, D. Angioni, G. Piras, G. Fumera, B. Biggio, and F. Roli, "Adversarial attacks against uncertainty quantification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2023, pp. 4599–4608.

[49] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "{CADE}: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2327–2344.

[50] B. Ganguly and V. Aggarwal, "Online federated learning via non-stationary detection and adaptation amidst concept drift," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 643–653, 2023.

[51] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification in the presence of concept drift," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 805–823.

[52] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.

[53] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer, 2005, vol. 29.

[54] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[55] J. Lin, R. Luley, and K. Xiong, "Active learning under malicious mislabeling and poisoning attacks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[56] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 45–48.

[57] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273–294.

[58] VirusTotal, "Virustotal: Free online virus, malware and url scanner," https://www.virustotal.com/, 2004, accessed: 2025-01-08.

[59] G. Han, J. Choi, H. G. Hong, and J. Kim, "Data poisoning attack aiming the vulnerability of continual learning," in *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2023, pp. 1905–1909.

[60] Z. Guo, A. Kumar, and R. Tourani, "Persistent backdoor attacks in continual learning," *arXiv preprint arXiv:2409.13864*, 2024.

[61] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[62] W. Zhao, J. Long, J. Yin, Z. Cai, and G. Xia, "Sampling attack against active learning in adversarial environment," in *Modeling Decisions for Artificial Intelligence: 9th International Conference, MDAI 2012, Girona, Catalonia, Spain, November 21-23, 2012. Proceedings 9*. Springer, 2012, pp. 222–233.

[63] B. Miller, A. Kantchelian, S. Afroz, R. Bachwani, E. Dauber, L. Huang, M. C. Tschantz, A. D. Joseph, and J. D. Tygar, "Adversarial active learning," in *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, 2014, pp. 3–14.

[64] G. Apruzzese, A. Fass, and F. Pierazzi, "When adversarial perturbations meet concept drift: an exploratory analysis on ml-nids," in *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*, 2024, pp. 149–160.

[65] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1332–1349.

[66] K. Zhao, H. Zhou, Y. Zhu, X. Zhan, K. Zhou, J. Li, L. Yu, W. Yuan, and X. Luo, "Structural attack against graph based android malware detection," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 3218–3235.

[67] P. He, Y. Xia, X. Zhang, and S. Ji, "Efficient query-based attack against ml-based android malware detection under zero knowledge setting," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 90–104.

[68] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera, "Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence," *Information fusion*, vol. 99, p. 101805, 2023.

[69] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[70] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[71] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.

[72] D. Watson, J. O' Hara, N. Tax, R. Mudd, and I. Guy, "Explaining predictive uncertainty with information theoretic shapley values," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 7330–7350. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/16e4be78e61a3897665fa01504e9f452-Paper-Conference.pdf

APPENDIX A
SUPPLEMENTARY CONTENT

A. Notation

Table IX presents the symbols used in the description of the PACDA.

TABLE IX: List of Symbols on Concept Drift Adaptation

Symbol	Description
$f_{\boldsymbol{\theta}_{n-1}}$	Victim model with parameters $\boldsymbol{\theta}_{n-1}$
$\boldsymbol{D}_{tr}^{n-1}$	Training dataset of victim model
$\boldsymbol{D}_{te}^{n}$	Test dataset during concept drift cycle n
$\boldsymbol{D}_{dr}^{n}$	Concept drift samples during concept drift cycle n
β	Label budget of the victim model
$\mathbf{x}_{tar}$	Attack target
C_n	Labeling budget consumption for the specific attack target
$\boldsymbol{D}_{seed}^{n}$	Poisoning attack seed dataset
$f_{\boldsymbol{\theta}_{n-1}^{*}}$	Surrogate model with parameters $\boldsymbol{\theta}_{n-1}^{*}$
$\boldsymbol{V}_{shap}$	Uncertainty feature attributions for test dataset
$\boldsymbol{D}_{\alpha}^{n}$	Poisoned samples generated by problem space perturbation
$\boldsymbol{D}_{shap}^{n}$	Poisoned samples generated by feature space perturbation

B. Shapley Additive Explanations

Recent advances in explainable machine learning have introduced various methods for interpreting the predictions of complex models [68]. SHapley Additive exPlanations [69] adopt the Shapley value concept from cooperative game theory to assign each feature a contribution score based on its influence on the model's output. The SHAP framework has been shown to subsume several earlier model explanation techniques [39], including LIME [70] and Integrated Gradients [71]. These model explanation frameworks help determine the importance of each feature value in the model's prediction.

$$g(\mathbf{x}) = \phi_0 + \sum_{j=1}^{M} \phi_j x_j \tag{15}$$

Explanation frameworks construct a surrogate linear model using the input feature vectors and output predictions, leveraging its coefficients to estimate feature importance and direction, as shown in Equation 15. $\mathbf{x}$ is the sample, x_j is the j^{th} feature for sample $\mathbf{x}$, and ϕ_j is the contribution of feature x_j to the model's prediction. The SHAP framework stands out by providing theoretical guarantees for calculating feature contributions in a model-agnostic manner. Recent studies have extended model prediction explanations to include explanations of prediction uncertainty [72]. Motivated by this, we incorporate SHAP-based uncertainty attribution into the design of our poisoning attack against concept drift adaptation methods.

C. Concept Drift Dataset Settings

Concept drift datasets have significantly advanced machine learning research due to their rich temporal attribute information. However, collecting natural concept drift datasets poses challenges, as it incurs significant costs and demands accurate temporal attributes. Therefore, synthetic datasets are often constructed for experimental evaluation in existing studies, in addition to real-world datasets.

1) Synthetic Dataset Construction: In this experimental evaluation, we constructed synthetic concept drift datasets using a publicly available spam email dataset (SPAM-Email [30]) and the MNIST image dataset [29].

- **MNIST:** We processed the MNIST handwritten digit dataset to simulate the concept drift phenomenon. Specifically, we selected two clusters of digits, close in the feature space, to represent two classes: digits 3, 5, and 8 (labeled as 0) and digits 4, 7, and 9 (labeled as 1). Each digit is treated as a subclass, with new subclasses gradually introduced based on feature similarity to simulate gradual concept drift. We randomly selected 30% of the data from digits 3 and 4 for the initial training set. The subsequent five cycles were used as testing periods, each selecting a portion of the unused data as the testing data.
- **SPAM-Email:** Similar to the concept drift construction in the MNIST dataset, we conducted an in-depth analysis of the original SPAM dataset. Using sklearn's k-means algorithm with default parameters, we subdivided legitimate and spam emails into 12 clusters, each representing a distinct subclass. Based on this, we designed a concept drift partitioning scheme with nine cycles, each containing 1,036 samples and an approximate 3:1 ratio of legitimate to spam emails. To simulate concept drift, we introduced a new subclass of legitimate and spam emails in each cycle while ensuring data from the same subclass appeared in consecutive cycles to reflect emerging trends. The first cycle served as the training data, containing one subclass for legitimate emails and one for spam. In subsequent testing cycles, new subclasses were introduced, and the proportions of existing subclasses were adjusted, ensuring at least four subclasses were present in each cycle. This approach enabled a smooth evolution of the data distribution over time while ensuring orderly family replacement.

D. Training and Testing Data Splits

Most training and testing data partitions are static, with the model trained on the training dataset and evaluated on a fixed testing data. However, testing data in concept drift scenarios are dynamic and continuously evolving. Both the distribution and size of the testing data change as time progresses.

- **APIGraph:** The APIGraph [27] dataset is trained on data from 2012, with concept drift testing conducted on data from 2013 to 2018, where the data for each year is incrementally released monthly. The ratio of legitimate to malicious software for each year is roughly maintained at 9:1, simulating the real-world scenario where legitimate samples significantly outnumber malicious ones. Detailed data partitioning is provided in Table III.
- **BODMAS:** The BODMAS [28] dataset comprises 57,293 malware samples and 77,142 benign samples. The malware samples were randomly selected monthly from a security company's internal malware database, and data collection lasted from August 29, 2019, to September 30, 2020.
  The data collected before October 2019 is used as the initial training data, while the data collected afterward serves as the concept drift testing data, as shown in Table X. The testing data is evaluated every month.

TABLE X: Windows Malware Concept Drift Dataset

Year	Malware	Benign	Malware Family
Train(before 10/19)	4,741	17,655	330
Test-10/19 (Cycle 1)	4,549	3,925	236
Test-11/19 (Cycle 2)	2,494	3,718	146
Test-12/19 (Cycle 3)	4,039	6,120	147
Test-01/20 (Cycle 4)	4,510	5,926	183
Test-02/20 (Cycle 5)	4,269	3,703	166
Test-03/20 (Cycle 6)	4,990	3,577	192
Test-04/20 (Cycle 7)	4,640	5,201	162
Test-05/20 (Cycle 8)	5,449	6,121	180
Test-06/20 (Cycle 9)	4,217	8,182	111
Test-07/20 (Cycle 10)	4,995	6,392	144
Test-08/20 (Cycle 11)	3,823	2,514	125
Test-09/20 (Cycle 12)	4,577	4,198	152
Total	**57,293**	**77,142**	**2,274**

- **SPAM-Email:** The dataset contains spam and legitimate messages, with a moderate spam ratio of approximately 25%. It includes 9,324 instances and 500 features. The initial training data consists of 771 legitimate emails and 265 spam emails. Subsequently, 265 new spam emails and 771 legitimate emails are introduced in each of the 8 concept drift testing cycles.

TABLE XI: Synthetic Concept Drift Dataset for SPAM-Email

miner	Spam	Legitimate
Train	265	771
Test-1 (Cycle 1)	265	771
Test-2 (Cycle 2)	265	771
Test-3 (Cycle 3)	265	771
Test-4 (Cycle 4)	265	771
Test-5 (Cycle 5)	265	771
Test-6 (Cycle 6)	265	771
Test-7 (Cycle 7)	265	771
Test-8 (Cycle 8)	265	771
Total	**2,387**	**6,937**

- **MNIST:** The MNIST dataset contains handwritten digits. 3,591 samples were used for training, and 31,860 samples were distributed across 5 concept drift testing cycles.

TABLE XII: Synthetic Concept Drift Dataset for MNIST

Year	Label-1	Label-0
Train	1,752	1,839
Test-1 (Cycle 1)	1,752	2,923
Test-2 (Cycle 2)	2,421	2,852
Test-3 (Cycle 3)	2,463	2,867
Test-4 (Cycle 4)	2,734	2,603
Test-5 (Cycle 5)	6,930	4,315
Total	**18,052**	**17,399**

E. Attack Target List

The experimental evaluation was conducted under various attack target configurations to validate the effectiveness of the proposed method and to analyze factors influencing attack success. The detailed information is as follows.

- **Single-Target Attack:** We followed two principles when selecting attack targets. First, the attack target must not be included in the training set. Second, the attack target is misclassified during the month it appears. This indicates that the attack target is a newly emerging concept drift sample in the testing phase rather than a simple modification of existing samples in the training data. Moreover, this also prevents situations where the attack target lacks sufficient attack value.
- **Multi-Target Attack:** The attack targets for multi-target attacks are composed of multiple single-attack targets that emerge simultaneously. The targets of APIGraph are detailed in Table XIII. The concept drift adaptation associated with the attack target spans five years. For other datasets, the multi-target attack is conducted over the entire set of attack targets.

TABLE XIII: Attack Target for Multi-target Attack

Month	Type	Family
2013-09	Non-Target	ansca cardserv svpeng
	Target	mecor smforw vietsms
2014-05	Non-Target	gabas simplocker smssend
	Target	mecor svpeng
2014-06	Non-Target	chyapo pletor spyware tebak
	Target	mecor adflex
2014-09	Non-Target	fobus gamecheater ransomware
	Target	mecor spyware
2014-10	Non-Target	fakebank systemmonitor webapp
	Target	airpush mecor
2015-05	Non-Target	adflex kalfere styricka
	Target	mobidash vnapstore
2016-07	Non-Target	clicks
	Target	adflex blouns mspy
2017-01	Non-Target	mobidash
	Target	batmob kalfere

F. Attack Target Initial Misclassification Time

Since the effectiveness of the PACDA is defined by prolonging the misclassification duration of the original attack target, it is essential to test the misclassification duration of the attack target in the absence of any attacks. We analyzed the misclassification time of malware under different labeling

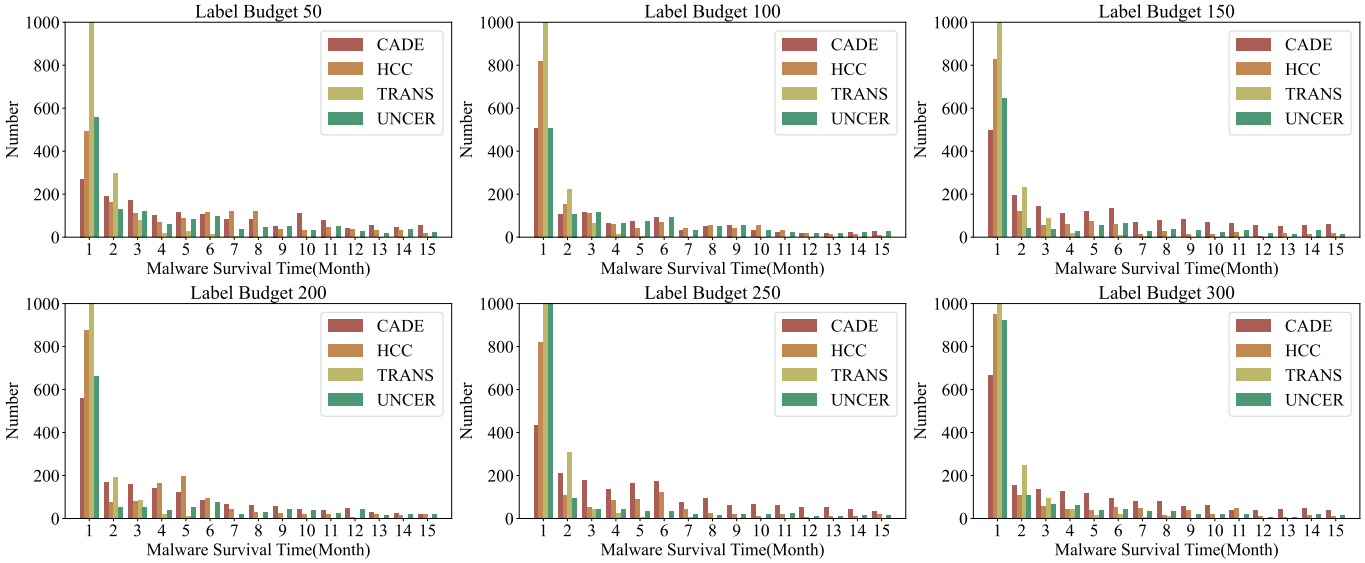

Fig. 10: Original misclassification time of attack targets

budget settings and concept drift adaptation strategies, as shown in Figure 10. Most malware is misclassified for 1-5 months under CDA-AL, while a small portion survives for over 5 months. To improve clarity, the bar chart for 0-2 months survival time samples in the TRANS statistics was truncated, maintaining the relative proportions of the overall sample volumes. The PACDA aims to extend the misclassification duration of targeted samples. Therefore, we selected all malware samples in the testing phase with a misclassification duration of 15 months or less as attack targets. We performed similar operations on the other three datasets to extract the original misclassification times of the attack targets.