PANDORA: Targeted Poisoning Attacks against Concept Drift Adaptation with Active Learning via Uncertainty Ranking Manipulation

Abstract—**Machine learning models often experience performance degradation over time due to concept drift. To address this, Concept Drift Adaptation with Active Learning (CDA-AL) adapts to evolving data distributions through selective labeling and retraining on the testing data stream. While existing poisoning attacks can disrupt CDA, targeted misclassification without degrading overall CDA-AL performance remains challenging. In this paper, we present PANDORA, a novel targeted poisoning attack against concept drift adaptation with active learning via uncertainty ranking manipulation. By injecting crafted high-uncertainty poisoned samples, PANDORA manipulates the victim model's uncertainty ranking on the testing data stream, misguiding the process of CDA-AL.**

We evaluate the effectiveness of PANDORA on five concept drift datasets from diverse domains, including Android and Windows malware detection, spam filtering, and image recognition. The attack results show that when the number of poisoned samples is limited to no more than 5% of the testing stream, PANDORA effectively extends the misclassification duration for over 80% of the attack targets (concept drift samples) by at least one concept drift cycle across four datasets. Our analysis further shows that existing countermeasures fail to defend against PANDORA, prompting us to propose an adaptive sample filtering method based on intra-cluster distance. The proposed defense successfully prevents the extension of misclassification duration for 40% of the attack targets under the same settings.

1. Introduction

Supervised machine learning has been widely applied to critical security applications, including intrusion detection [1] and malware analysis [2], where accurate and timely classification is essential for preventing breaches and mitigating threats. However, as the gap between the testing and training data distributions widens over time [3], machine learning models performance on the testing data declines [4]. This phenomenon, known as concept drift [5]–[7], is particularly dangerous because it can result in inaccurate predictions and potentially critical failures in high-stakes applications.

To address this challenge, numerous Concept Drift Adaptation (CDA) approaches have been proposed [8]–[10]. Among them, Concept Drift Adaptation with Active Learning (CDA-AL) [11]–[13] has emerged as a promising strategy. In CDA-AL, the model selectively queries labels for the most informative data samples—typically those with the highest prediction uncertainty [11], [14]—to detect and adapt to concept drift. These selected samples are manually labeled and incorporated into the training dataset for model retraining. As a result, CDA-AL improves the model's prediction accuracy on evolving data distributions.

Because CDA-AL continuously collects testing data stream and selects retraining samples from it, the framework may become vulnerable to poisoning attacks. However, unlike traditional poisoning attacks that inject malicious samples directly into the training dataset in a batch manner [15]–[19], attacks against CDA-AL must exploit its adaptive learning process by planting poisoned samples in the testing data stream. This is a challenging task, as attackers must deceive the retraining process into mistakenly selecting poisoned samples for labeling, which are then incorporated into the retraining dataset—a scenario that, to the best of our knowledge, has not been explored before.

The closest prior work is poisoning attacks or adversarial perturbations on CDA [20], [21]; however, this area remains largely underexplored. These studies typically assume that poisoned samples from the testing data stream are guaranteed to be incorporated into the retraining dataset, which does not align with the selective sample acquisition mechanism used in CDA-AL. Moreover, their focus is primarily on untargeted attacks aimed at degrading overall model performance, overlooking the design of targeted poisoning strategies that can preserve stealth and global accuracy while manipulating specific predictions.

In this paper, we present PANDORA, a novel targeted poisoning attack against concept drift adaptation with active learning via uncertainty ranking manipulation. PANDORA treats newly emerging concept drift samples in the testing stream as attack targets, aiming to prolong their misclassification while maintaining the overall performance of the victim model. Specifically, PANDORA generates high-uncertainty poisoned samples via adversarial perturbations. These samples manipulate the uncertainty ranking in the testing data stream of each concept drift cycle, increasing their chances of being selected during active learning-based retraining. The poisoned samples ultimately deplete the victim model's limited labeling budget, thereby hindering effective learning on concept drift samples associated with the attack targets.

We evaluated the effectiveness of PANDORA on five concept drift datasets: APIGraph [22] , Androzoo [37] and BODMAS [23] for malware detection, MNIST [24] for

image recognition, and SPAM [25] for spam filtering. Even when the number of poisoned samples is limited to no more than 0.05 of the testing stream, PANDORA effectively extends the misclassification duration for over 80% of the attack targets by at least one concept drift cycle across four datasets. To better understand the effectiveness of PANDORA 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 methods for constructing sample features. Notably, PANDORA remains effective across all settings, and we provide an in-depth analysis of the relationship between influencing factors and attack performance.

We also evaluated the effectiveness of existing targeted poisoning defense techniques against PANDORA, including data sanitization [26], robust training [27], and input-time detection [28]. This corresponds to the three stages of potential defense methods for CDA-AL: data processing, sample selection, and model retraining. Experimental results indicate that these methods struggle to effectively distinguish between poisoned and clean samples, leading to the failure of defenses against PANDORA. These findings highlight the urgent need for new defense mechanisms against PANDORA. To address this, we propose a poisoned sample filtering approach based on intra-cluster distance to mitigate the effects of PANDORA. Under the same poisoning attack settings, our defense method successfully prevents the extension of misclassification duration for 40% of the attack targets. The contributions of this paper are as follows:

- We present PANDORA, the first targeted poisoning attack against CDA-AL. PANDORA causes the misallocation of labeling budget to poisoned samples by manipulating the uncertainty ranking of testing data, leading to misclassification of attack targets during the adaptation process.
- We introduce a strategy for generating high-uncertainty poisoned samples. By applying problem-space adversarial perturbations and concept drift direction misguidance, our method generates enough poisoned samples to exhaust the limited labeling budget.
- We evaluate PANDORA across diverse datasets, model architectures, and sample selection strategies, demonstrating its effectiveness in prolonging the misclassification duration of attack targets. To demonstrate our commitment to open-source, we have made an anonymized version of the code available at https://anonymous.4open.science/r/PANDORA-Attack-B730
- We identify critical limitations in existing defense mechanisms against PANDORA. We present a poisoned sample filtering method based on cluster-adaptive distance to mitigate this attack.

2. Background: Concept Drift Adaptation with Active Learning

CDA-AL is a paradigm that enhances model performance by identifying high-uncertainty samples from the testing data [3], [11], [29]. 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 [11]: $\mathbf{u}_i = 1 - \max(\mathbf{c}_i, 1 - \mathbf{c}_i)$. 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.

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

3. Threat Model

The goal of PANDORA is to ensure persistent misclassification of the attack targets by the victim model throughout the CDA-AL process. We assume attackers cannot access the victim model's internal parameters or influence its training process [30], including manual labeling of CDA-AL [11]. Attackers can only submit samples to the victim model for querying to obtain outputs such as sample uncertainty scores [31] and predicted labels [32]. Additionally, attackers are presumed to have access to public data and the resources required to train surrogate models [33]. Consistent with previous research on active learning attacks [34], we assume the attacker typically has knowledge of the victim model's labeling budget settings. Moreover, the attacker can access the victim model's prediction uncertainty. This is based on the observation that existing machine learning services (e.g., image recognition [31]) provide uncertainty information to help users better interpret and utilize the prediction results. Similarly, malware detection services such as VirusTotal [32] provide outputs from multiple detection models, enabling users to estimate prediction uncertainty

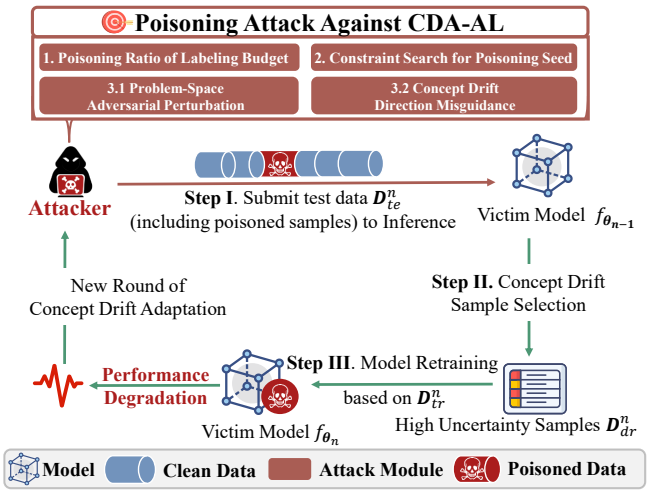

Figure 1: Poisoning Attack against CDA-AL

based on the distribution of these results. In addition, it is important to note that although CDA-AL systems involve human analysts, their primary role is to provide reliable labels for concept drift samples [11]. As long as PANDORA does not rely on malicious label manipulation, the attack remains inconspicuous to human analysts.

4. PANDORA

We present the PANDORA framework, illustrated in Figure 1. The process begins by estimating the required poisoning ratio of the labeling budget based on the specified attack targets. By injecting poisoned samples into the testing data stream, the attacker determines which knowledge the victim model can or cannot acquire. Subsequently, we search the testing data stream to identify potential poisoning seeds. Based on the identified poisoning seeds, PANDORA generates poisoned samples through adversarial perturbations in the problem space, further enhancing the attack effectiveness by misleading the direction of concept drift. Ultimately, PANDORA exhausts the victim model's labeling budget, preventing effective learning of the attack target.

We demonstrate the design of the PANDORA method by using a specific 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 concept drift sample 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).

4.1. Poisoning Ratio of Labeling Budget

Once the attack target $\mathbf{x}_{tar}$ is determined, the PANDORA begins by estimating the poisoning ratio 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 $\mathbf{x}_{tar}$. If the uncertainty ranking r_{tar} of the attack target $\mathbf{x}_{tar}$ is higher than the labeling budget boundary ($r_{tar} < r_\beta$), the attack target $\mathbf{x}_{tar}$ 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 $\mathbf{x}_{tar}$ will be learned during the retraining process of the victim model. The gap between the uncertainty ranking of the attack target $\mathbf{x}_{tar}$ 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 $\mathbf{x}_{tar}$ 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:

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

Furthermore, considering the inherent randomness in sample selection and model training, we introduce an amplification factor to the originally computed label budget consumption C_n to mitigate the effects of randomness. The specific value of this factor is determined by the attacker's computational capabilities.

4.2. Constraint Search for Poisoning Seed

The attacker then searches for 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 key issue is ensuring that the poisoned samples have high uncertainty, thereby increasing the likelihood that they consume the victim model's labeling budget. So the uncertainty of these poisoned seed

samples must exceed the attack target's uncertainty, as shown in Equation 3.

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

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 $\mathbf{x}_{tar}$. Therefore, the attacker must exclude malware samples from the poisoning seed data to avoid reducing the attack effectiveness on the target $\mathbf{x}_{tar}$. In addition, similarity-based filtering can be employed to prevent the inclusion of samples similar to the attack target $\mathbf{x}_{tar}$ in the poisoned samples.

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

Each row of the poisoning seed data $\boldsymbol{D}^n_{seed}$ is denoted as a sample vector $\mathbf{x}_j$, where $\mathbf{x}_j = \boldsymbol{D}^n_{\text{seed}}[j,:], \forall j \in \{0, \ldots, K-1\}$. 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 $\boldsymbol{D}^n_{seed}$. Finally, the refined poisoning seed data is sorted in descending order of uncertainty, with the sample at index 0 having the highest uncertainty.

4.3. Poisoned Sample Generation

The poisoned seed samples $\boldsymbol{D}^n_{seed}$ is incorporated into the victim model's training dataset 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. Consequently, attack targets will still be selected and labeled manually, leading to a failure of PANDORA. To ensure the attack effectiveness, 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. The poisoned sample generation process consists of two parts. The first part generates poisoned samples based on high-uncertainty poisoning seeds to exhaust the labeling budget, thereby preventing the victim model from learning samples directly related to the attack targets. The second part constructs poisoned samples using shapley additive explanations to misguide the direction of concept drift, with the aim of eliminating samples that are implicitly related to the attack targets within the labeling budget. The combination of the two parts prevents the victim model from effectively learning the attack targets while also hindering its understanding of the correct direction of concept drift.

4.3.1. Problem-Space Adversarial Perturbation. Adversarial perturbation is particularly effective for rapidly and cost-efficiently adjusting the uncertainty rankings of the testing data. 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 these 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 dataset denoted as $\boldsymbol{D}^n_{\alpha}$. α denotes the problem-space perturbation operations. 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 boundary (i.e., $r_{tar} > r_{\beta}$).

TABLE 1: Problem-Space Perturbation Operations

Dataset	Perturbation Operations (α)
APIGraph	Rename method names to meaningless identifiers
BODMAS	Dynamically adjust the size of the DOS STUB space
MNIST	Apply sharpening to enhance the edges of the image
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 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 1. 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 C_n.

4.3.2. Concept Drift Direction Misguidance. The uncertainty of poisoned samples generated through problem-space adversarial 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 includes samples that affect the misclassification of the attack targets (e.g., malware sharing similar vulnerabilities with the target), the attack effectiveness may be reduced. Since these high-uncertainty concept drift samples are beyond the control of the attacker, we need to construct poisoned samples with higher uncertainty than the poisoned seed samples to mislead the victim model in its learning of the direction of concept drift. Building on the insights of Ledda et al., who employed perturbation search techniques to identify minimal perturbations that alter a sample's uncertainty [36], we aim to manipulate the uncertainty of poisoned samples. However, their method focuses on reducing uncertainty, whereas CDA-AL selects samples with high uncertainty. To overcome this limitation, we first conduct uncertainty attribution on the samples to identify features that can increase their uncertainty. Given the need to accommodate various models for CDA-AL, our uncertainty attribution method should be model-agnostic. Moreover, to effectively construct poisoned samples with high uncertainty, the explanation should operate at the feature level. Although standard conformal prediction methods can provide interpretable uncertainty estimates through credibility scores, they are unable to offer feature-level explanations of uncertainty. Therefore, we adopt a shapley additive explanations (SHAP) based approach for feature level attribution of sample uncertainty. The fundamental concepts of SHAP are provided in Appendix A.2. We employ a standardized permutation-based SHAP method to interpret sample uncertainty, ultimately getting an uncertainty attribution matrix for the samples.

In selecting the analysis targets for uncertainty attribution, it is important to note that since the objective of PANDORA is to prevent the victim model from learning the attack target $\mathbf{x}_{tar}$, the generated poisoned samples must hinder the model's ability to adapt to concept drift. To achieve this, samples outside the labeling budget are selected as targets for uncertainty attribution. Their lower importance relative to in-budget samples helps avoid inadvertently strengthening the model's adaptation capabilities. the testing data outside the labeling budget, denoted as $\boldsymbol{D}^n_{shap}$ ($\boldsymbol{D}^n_{shap} = \boldsymbol{D}^n_{te} \setminus \boldsymbol{D}^n_{dr}$). Then We compute the uncertainty attribution matrix of $\boldsymbol{D}^n_{shap}$, as shown in Equation 5.

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

$\boldsymbol{V}_{shap}$ represents the matrix of uncertainty feature attributions for data $\boldsymbol{D}^n_{shap}$. 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.

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

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 $\boldsymbol{D}^n_{shap}$. The attacker can further amplify the set by applying problem space perturbation to the poisoned samples $\boldsymbol{D}^n_{shap}$ to enhance their impact on the victim model.

5. Evaluation

5.1. 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 PANDORA evaluation is conducted on five datasets: two synthetic datasets (MNIST [24] and Spam-Email [25]) and three real-world datasets (APIGraph [22], Androzoo [37] and BODMAS [23]). Detailed dataset information is presented in

TABLE 2: Concept Drift Datasets for Attack Evaluation

Dataset	Type	Duration	Size
APIGraph [22]	Android Malware	7 years	320,315
Androzoo [37]	Android Malware	6 years	265,740
BODMAS [23]	Windows Malware	2 years	149,217
MNIST [24]	Image Classification	5 cycles	70,000
SPAM [25]	Spam Emails	8 cycles	9324

Table 2. 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 [39]. Synthetic dataset creation details are in Appendix A.3.1.

In addition, concept drift datasets require the testing data to be partitioned according to the order of occurrence. Following prior work [11], we adopt a similar setup. For datasets with timestamps, the APIGraph [22] dataset serves as an example: data from 2012 is used for training, while data from 2013 to 2018 is used for testing, as illustrated in Table 3. In the subsequent concept drift experiments,

TABLE 3: 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

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 information is provided in Appendix A.4. Additionally, the Androzoo and APIgraph datasets represent similar scenarios and are primarily used for analyzing the impact of feature selection. Therefore, they are discussed in detail in Section 5.3.3.

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 [22], 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 [11], [35], [40], [41]. 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. For the sample selection strategy, we use the classic confidence-based method [41]. To prevent overfitting during model training, we adopted the same parameter settings as those used in existing concept drift adaptation methods [11] Additionally, we incorporated a dropout mechanism in the model trained on the BODMAS dataset, and employed early stopping strategies for models trained on the other three datasets to further mitigate overfitting. Moreover, as shown in Table 5, the victim models achieved an average accuracy of 97.94% and an average F1 score of 0.88 on the testing data, indicating no signs of overfitting. Other parameters is shown in Appendix A.5

Attack Targets: 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. Appendix A.6 provides detailed attack targets information. In the real world, multiple attack targets may exist at the same concept drift cycle. Therefore, we classify the evaluation of attack effectiveness into two types: single-target attack and multi-target attack, as shown in Section 5.2. This setup aims to understand better the impact of different attack targets settings on attack effectiveness. It is worth noting that the multi-target attack setting simulates both a single attacker targeting multiple attack targets and multiple attackers independently targeting different targets simultaneously.

Attack Baselines: Given the absence of targeted poisoning attacks tailored for CDA-AL, we adapt representative adversarial concept drift poisoning attack methods to serve as baselines for comparison.

- Blind Pertubation: Apruzzese et al. [21] proposed a blind perturbation attack in the problem space that degrades concept drift adaptation performance in intrusion detection scenarios. Following this approach, we randomly select β (labeling budget) samples from the concept drift testing data as perturbation targets and inject them into the testing data stream.
- Outdate Concept: Korycki et al. [20] proposed a direction-shifting attack to mislead the concept drift adaptation process. Given that the direction of real-world concept drift is often unpredictable, we adopt a strategy that replays samples from the training dataset to mislead the victim model and degrade its adaptation performance.

Metrics: The effectiveness of PANDORA 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 specific 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 PANDORA prolongs the duration of the attack target's misclassification will the attack be regarded as successful, as defined in Equation 8. N denotes the PANDORA 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.

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

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

5.2. Attack Effectiveness

The effectiveness of PANDORA is evaluated on five datasets across different domains. Due to its long time span and real-world dataset, the APIGraph dataset [22] is utilized for both single-target and multi-target attacks. The manually synthesized concept drift datasets (MNIST [24] and SPAM [25]), as well as the BODMAS malware concept drift dataset [23], contain a limited number of attack targets and span a short time period. Therefore, they are utilized to evaluate multi-target attack scenarios.

5.2.1. Single-Target Attack. We assess the effectiveness of the PANDORA with a simple attack target on the API-Graph [22] dataset. The evaluation of attack effectiveness involves more than 300 attack targets. Appendix A.6 provides detailed information on the attack targets.

TABLE 4: Effectiveness of Different Attack Strategies

Attack Strategies	F1 Score	ACC	ASR (%)
Blind Pertubation [21]	0.92	0.98	38.16%
Outdate Concept [20]	0.92	0.98	34.31%
PANDORA (Our Attack)	0.90	0.98	89.22%

We first apply problem-space adversarial perturbation to generate poisoned samples. Subsequently, by employing concept drift direction misguidance, 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. To efficiently demonstrate the effectiveness of PANDORA, we compare it with two existing adversarial concept drift poisoning attack methods, as shown in Table 4. The PANDORA demonstrates high effectiveness, achieving an average ASR of 89.22%. This implies that the PANDORA can extend the misclassification duration of most attack targets during the process of CDA-AL. In addition, compared to existing attack methods, PANDORA significantly improves the attack success rate, with an average increase of 52.99%. Moreover, the overall performance of the victim model remains stable, with an F1 score of 0.9 and an accuracy of 98%.

In Table 5, 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 60% of the attack targets families achieve an ASR of 100%, we focus on whether the impact of the PANDORA 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 PANDORA 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 adversarial perturbation effectively prevents the victim model from learning the attack targets

TABLE 5: 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)	80.00%	0.90 (-0.02)	98.26 (-0.30)
Smforw (Trojan-Spy)	100%	0.81 (-0.11)	96.80 (-1.76)
Fobus (Data Stealing)	42.86%	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)	60%	0.90 (-0.02)	98.40 (-0.16)
Mogap (Trojan-Spy)	100%	0.90 (-0.02)	98.24 (-0.32)
Congur (Trojan-Spy)	87.50%	0.91 (-0.01)	98.40 (-0.16)

The proportion of poisoned samples in the monthly testing data averages less than 0.06, demonstrating a high level of attack stealth.

while maintaining its overall performance during the CDA-AL. 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, as shown in Table 6. In addition, the process can be supported by a range of well-established tools, highlighting the cost-effectiveness of problem-space perturbation.

TABLE 6: APK obfuscation Time

APK	Size (MB)	Obfuscation time
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
Average	199.72	90.72s

Since the victim model is continuously updated during the concept drift adaptation process, it is necessary to test whether PANDORA'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 of misclassification, based on the initial misclassification results (as shown in Appendix A.7). 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 PANDORA 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. During the six-year concept drift adaptation process, an average Attack Success Rate (ASR) of 87.33% was achieved, as shown in Figure 2. Therefore, it is evident that the PANDORA ensures the attack targets 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.

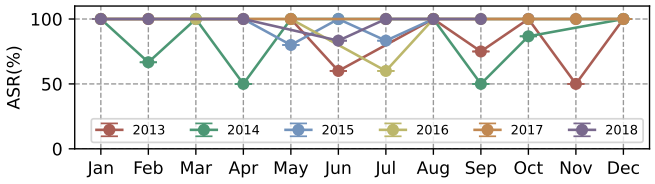

Figure 2: Attack Persistence over 6 years

The effectiveness of the PANDORA using the problem-space adversarial perturbation in poisoned sample generation has already been validated. Next, we evaluate the effectiveness of the concept drift direction misguidance. We selected 138 families with less than six months of survival for comparative analysis. 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 concept drift misguidance based on

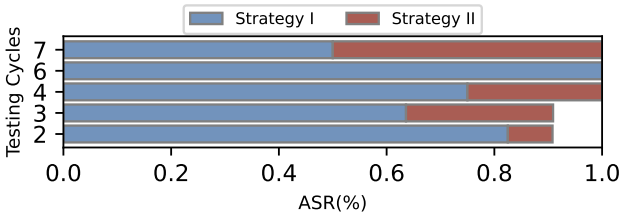

Figure 3: Feature-space perturbation strategy

the shapley additive explanations, 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 adversarial perturbation strategy. After applying concept drift misguidance, the average uncertainty ranking of the poisoned samples increased by 66.2%. This further demonstrates the complementary roles of the two modules in the generation of poisoned samples by PANDORA.

Additionally, we further tested the impact on the PANDORA 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 [11], 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.

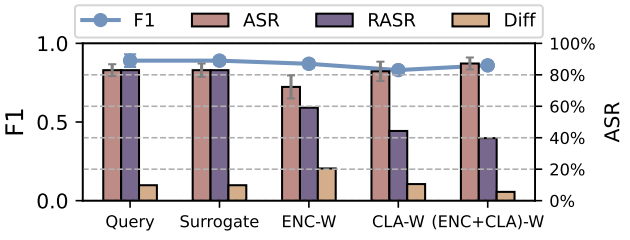

Figure 4: Attack Effectiveness under Weakened Capabilities

Previous studies have shown that when the attacker's capabilities are limited, the effectiveness of poisoning at-

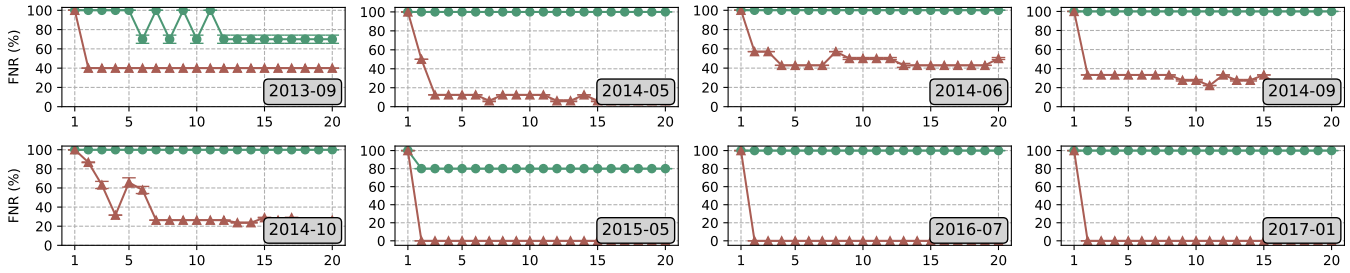

Figure 5: Attack Effectiveness with Multi-Target Attack (6 year Attack Testing)

tacks decreases [15]. However, PANDORA 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.

5.2.2. Multi-Target (Attacker) Attack. As described in Section 5.1, the multi-targets setting simulates two practical scenarios: (1) a single attacker launching attacks against multiple targets, and (2) multiple attackers independently targeting different targets. For clarity and ease of analysis, we adopt the first scenario, where a single attacker conducts multi-target attacks, as the basis for the following discussion. Multiple attack targets may emerge within the same concept drift cycle in real-world scenarios. Therefore, PANDORA also supports a multi-target attack mode. In the multi-target attack setting, the attacker performs PANDORA on multiple attack targets simultaneously. To evaluate the effectiveness of the PANDORA 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.6. The multi-target attack achieved an average success rate of 97.5% across different testing times, demonstrating its effectiveness in executing the PANDORA 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 PANDORA incurs a low cost, as a single attack benefits multiple targets simultaneously.

Then we discuss the scenario where multiple distinct attackers simultaneously target different attack targets. In terms of attack effectiveness, the success rate in the multi-attacker setting is consistent with that in the single attacker setting with multiple attack targets. This is because, as previously analyzed, the poisoned samples generated for different attack targets do not interfere with one another. In fact, poisoned samples crafted for attack targets with the highest uncertainty can even facilitate the misclassification of other attack targets. Furthermore, we observe that when attack effects are mutually beneficial, attackers tend to cooperate, thereby reducing the overall attack cost. Specifically, multiple attackers can identify the attack target with the highest uncertainty and use it as a common target, allowing them to share the costs involved in poisoned sample generation.

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 PANDORA on all attack targets at different concept drift time points. The average attack success rate reaches 81.98% (as shown in Table 7), with a notably high success rate of 92.92% achieved on the real-world concept drift dataset, APIGraph. Moreover, we observe that the vic-

TABLE 7: Multi-Target Attack across Four Ddatasets

Datasets	Targets	F1-Testing	F1-Validation	ASR (%)
APIGraph	331	$0.76_{-0.15}$	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%

tim 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, PANDORA stealth remains unaffected in both cases, as the testing data is unlabeled, and the victim model cannot promptly detect its performance degradation.

5.3. Attack Influencing Factors

We have demonstrated the effectiveness of PANDORA. To further investigate how real-world conditions affect attack performance, we analyze the factors influencing PANDORA using a real-world Android malware dataset (APIGraph [22]) spanning seven years.

5.3.1. Impact of Different CDA-AL Strategies. Existing concept drift adaptation strategies in sensitive domains [11], [35], [40] can be broadly categorized into four types. In addition to the uncertainty-based strategies discussed in Section 2, the remaining approaches fall into three categories. To ensure the generality of our findings, we evaluate the effectiveness of PANDORA across all these representative adaptation strategies.

- CADE [35] 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 [40] applies conformal prediction [42] to concept drift adaptation. It calculates a testing sample's non-conformity score, credibility (proportion of calibration samples with higher scores), and confidence (1 minus the opposite label's credibility). Low scores indicate potential drift.
- HCL [11] 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.

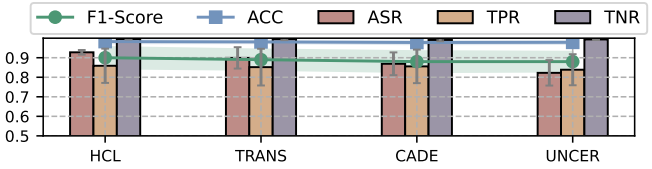

Figure 6: Different Concept Drift Adaptation Strategies

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

5.3.2. 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 [11]. 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, PANDORA remains effective across

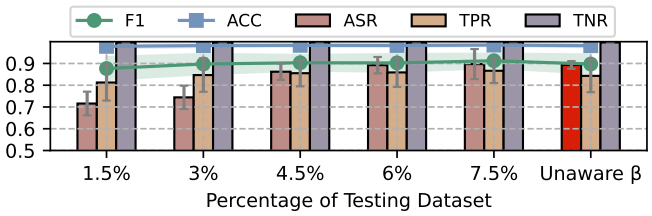

Figure 7: PANDORA Under Different Labeling Budget

budgets, with an average attack success rate of 85.10%.

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 [43] attacks, where the attacker maximizes attack traffic without knowing the victim's total resources to achieve the attack objective. Moreover, even if the scale of poisoned samples in the PANDORA attack exceeds the labeling budget, it is unlikely to raise suspicion. This is because the poisoned samples are injected into the unlabeled testing data stream during each concept drift cycle, and the volume of this stream is substantially larger than the victim model's monthly labeling budget. As a result, PANDORA exhibits strong stealth. Furthermore, it is extremely difficult for the victim model to detect poisoned samples within the testing data stream due to the massive volume of incoming data, which makes full inspection infeasible.

5.3.3. Impact of Feature Selection. Both APIGraph and Androzoo are Android malware datasets designed to capture concept drift. By comparing the differences in attack performance between these two datasets, we analyze how variations in feature construction affect the attack effectiveness. Since APIGraph employs a more advanced feature extraction method, we selected the classic DREBIN [38] approach for feature extraction on the Androzoo dataset. Experiments show that PANDORA still achieves strong performance, with an attack success rate over 90%. Moreover, compared to the no-attack setting, the victim model shows only a 0.01 change in F1 score and less than 1.5% variation in FNR, indicating strong stealthiness of the attack. The ability of PANDORA to achieve a high attack success rate under various feature selection methods demonstrates the robustness and generalizability of our attack strategy.

5.4. Ablation Study

To effectively demonstrate the importance of each component in PANDORA, we conduct ablation studies by removing key elements of the attack pipeline. This allows us to evaluate the necessity of each component and analyze the underlying factors contributing to PANDORA 's success. As described in Section 4, the PANDORA framework consists of three main components. Our ablation study focuses on analyzing the impact of these three core components. We conduct the ablation study on the APIGraph concept drift dataset [22], which features the longest duration of concept drift. As shown in Table 8, when the attacker lacks poisoning ratio analysis, they cannot determine the number of required poisoned samples. As a result, seed samples are directly injected into the testing stream. Due to the limited number of poisoned samples, the labeling budget is not effectively consumed, reducing the attack success rate to 10.87%. Disabling the poisoned seed search forces the attacker to randomly perturb testing samples, resulting in poisoned data with low uncertainty. This weakens labeling budget consumption, yielding an attack success rate of 10.99%. Finally, without the poisoned sample generation module, the attacker fails to produce samples aligned with labeling budget consumption requirements, again causing a drop in success rate to 10.87%.

TABLE 8: Attack value assessment necessity analysis

Ablation Component	Ablation-ASR (%)	ASR (%)
Poisoning Ratio Estimation	10.87	$89.22_{+78.35}$
Constraint Search for Poisoning Seed	10.99	$89.22_{+78.93}$
Poisoned Sample Generation	10.87	$89.22_{+78.35}$

6. Potential Defenses

CDA-AL methods lack directly relevant and effective defenses against poisoning attacks. While Lin et al. [44] 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 PANDORA against three representative targeted poisoning attack defense mechanisms [26]–[28]. The three methods correspond to the three possible stages at which defenses can be deployed within the CDA-AL framework.

- **Data Sanitization (Data Processing)**: activation clustering [26] (AC) 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.
- **Input-Time Detection (Sample Selection)**: Liu et al. [28] observed that targeted poisoning attacks often embed triggers into attack targets to induce misclassification. However, such triggered samples exhibit consistent corruption robustness (CRC) under input perturbations. Leveraging this observation, they proposed detecting targeted poisoning attacks by analyzing robustness variations.
- **Robust Training (Model Retraining)**: Data-Free Pruning [27] (DP) defends against data poisoning attacks by pruning neurons that remain dormant for clean data, based on the assumption that poisoned and clean samples activate different subsets of neurons.

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 PANDORA'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). As a result, existing defenses cannot distinguish between clean and poisoned samples, so they cannot fully defend against PANDORA attacks. Thus, the

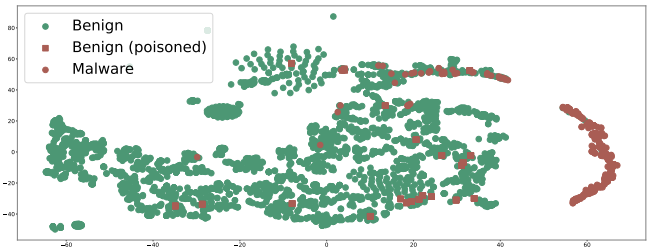

Figure 8: ICDF Defense Motivation

attack success 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 PANDORA.

Intra-Cluster Distance Filtering (ICDF): In the PANDORA, 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

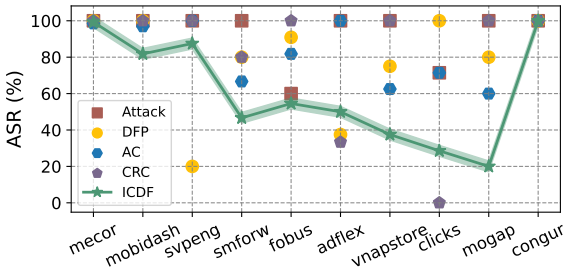

Figure 9: ICDF Defense (Single-Target PANDORA)

single-target attack mode, reducing the success rate of PANDORA by 39% across the Top 10 target sets, as shown in Figure 9. While other defense methods demonstrated some effectiveness, they were 20.29%, 12.43% and 22.43% less effective than ICDF, respectively. We observed that, with the exception of ICDF, which consistently reduced the attack success rate across all attack targets, other defense methods inadvertently increased the attack success rate for certain targets. Moreover, ICDF effectively reduces the attack success rate of PANDORA while causing only a 0.03 drop in F1-score compared to the no-attack scenario, thereby maintaining stable overall concept drift adaptation performance. We also conducted defensive testing under an enhanced attack strategy that introduces misleading concept drift. 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 9. The attack success rate decreased by an average of 54.97% across the four datasets. Therefore, our ICDF defense method also demonstrates strong effectiveness in defending against multi-target attack scenarios. Moreover, after introducing the ICDF defense method, the F1 score on the four datasets was restored to an average of 0.91, indicating that the normal concept drift adaptation performance was not affected.

TABLE 9: ICDF Defense (Multi-Target PANDORA)

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.

7. Limitation and Discussion

Existence of Attack Seeds. PANDORA 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, PANDORA remains feasible even when seeds are temporarily unavailable.

Completeness of Testing Data Collection. To launch PANDORA, 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 4.1. Furthermore, in real-world scenarios, attackers can leverage publicly available threat intelligence platforms, such as VirusTotal [32], 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 PANDORA.

Use Cases and Limitations. PANDORA is primarily designed for adversarial concept drift scenarios in the security domain (e.g., malware or spam detection), where misclassified attack targets still deliver expected functionalities, making the attack less likely to be noticed by end users. While PANDORA also achieves high attack success in benign drift settings (e.g., image recognition, as shown on the MNIST dataset), the persistent and noticeable misclassifications are more likely to trigger user complaints, thereby reducing attack stealthiness.

Our evaluation of attack effectiveness primarily focuses on the APIGraph [22] 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 [11], [35], [39].

Future Work. Another important future direction is to extend our research to the security of continual learning [45], [46]. 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 [47], 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. Moreover, targeted poisoning attacks in the context of CDA-AL remain underexplored, resulting in a limited number of effective defense strategies—particularly in security domains such as malware detection. As future work, we aim to further enhance the robustness of CDA-AL against targeted poisoning attacks.

8. Related Work

Poisoning Attacks Against Active Learning. Zhao et al. [48] 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. [49] 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. [34] 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 PANDORA requires no modifications to the attack targets.

Adversarial Concept Drift. Korycki et al. [20] 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 dataset, which contrasts with our setting, where only selected samples are labeled and used for training in CDA-AL. Apruzzese et al. [21] 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.

9. 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 PANDORA, 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] E. Yu, J. Lu, B. Zhang, and G. Zhang, "Online boosting adaptive learning under concept drift for multistream classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 16 522–16 530.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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 Computing Surveys*, vol. 55, no. 7, pp. 1–36, 2022.

[16] C. Gong, Z. Yang, Y. Bai, J. He, J. Shi, K. Li, A. Sinha, B. Xu, X. Hou, D. Lo *et al.*, "Baffle: Hiding backdoors in offline reinforcement learning datasets," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2086–2104.

[17] 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.

[18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 19–35.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] X. Liu, M. Li, H. Wang, S. Hu, D. Ye, H. Jin, L. Wu, and C. Xiao, "Detecting backdoors during the inference stage based on corruption robustness consistency," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 363–16 372.

[29] 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.

[30] 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*, 2017, pp. 506–519.

[31] Baidu, "Universal object and scene recognition," https://ai.baidu.com/tech/imagerecognition/general, 2025.

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

[33] Y. Qin, Y. Xiong, J. Yi, and C.-J. Hsieh, "Training meta-surrogate model for transferable adversarial attack," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 9516–9524.

[34] 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.

[35] 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.

[36] 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.

[37] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proceedings of the 13th international conference on mining software repositories*, 2016, pp. 468–471.

[38] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[39] 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.

[40] 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.

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

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

[58] D. Watson, J. O'Hara, N. Tax, R. Mudd, and I. Guy, "Explaining predictive uncertainty with information theoretic shapley values," *Advances in Neural Information Processing Systems*, vol. 36, pp. 7330–7350, 2023.

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 [50]–[52], we will restrict code sharing to verified academic researchers.

Appendix A. Supplementary Content

A.1. Notation

Table 10 presents the symbols used in the description of the PANDORA.

TABLE 10: 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
$\mathbf{x}_{tar}$	A specific 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

A.2. Shapley Additive Explanations

Recent advances in explainable machine learning have introduced various methods for interpreting the predictions of complex models [53]. SHapley Additive exPlanations [54] 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 [55], including LIME [56] and Integrated Gradients [57]. 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 [58]. Motivated by this, we incorporate SHAP-based uncertainty attribution into the design of our poisoning attack against concept drift adaptation methods.

A.3. 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.

A.3.1. Synthetic Dataset Construction. In this experimental evaluation, we constructed synthetic concept drift datasets using a publicly available spam email dataset (SPAM-Email [25]) and the MNIST image dataset [24].

- **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.

A.4. 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 [22] 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 3.
- **Androzoo:** The Androzoo [37] dataset is trained on data from 2017, with concept drift testing conducted on data from 2018 to 2022, where the data for each year is incrementally released monthly. The experimental settings for the other datasets are similar to those used for the APIGraph [22] dataset. Detailed data partitioning is provided in Table 11.

TABLE 11: Android Concept Drift Dataset (Androzoo)

Year	Malware	Benign	Malware Family
Train-2017	2,108	18,972	192
Test-2018 (Cycle 1)	4,625	41,625	363
Test-2019 (Cycle 2)	8,612	77,508	354
Test-2020 (Cycle 3)	3,512	31,608	283
Test-2021 (Cycle 4)	4,903	44,127	256
Test-2022 (Cycle 5)	2,814	25,326	136
Total	26,574	239,166	1,584

- **BODMAS:** The BODMAS [23] 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.

TABLE 12: 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**

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 12. The testing data is evaluated every month.

- **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 13: 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.

A.5. Model parameters

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. A learning rate decay factor of 0.05 was applied every 10 epochs starting from epoch 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.

TABLE 14: 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**

A.6. Attack Targets List

The experimental evaluation was conducted under various attack targets 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 targets must not be included in the training set. Second, the attack targets 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 targets lack 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 15. 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.

A.7. Attack Targets Initial Misclassification Time

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

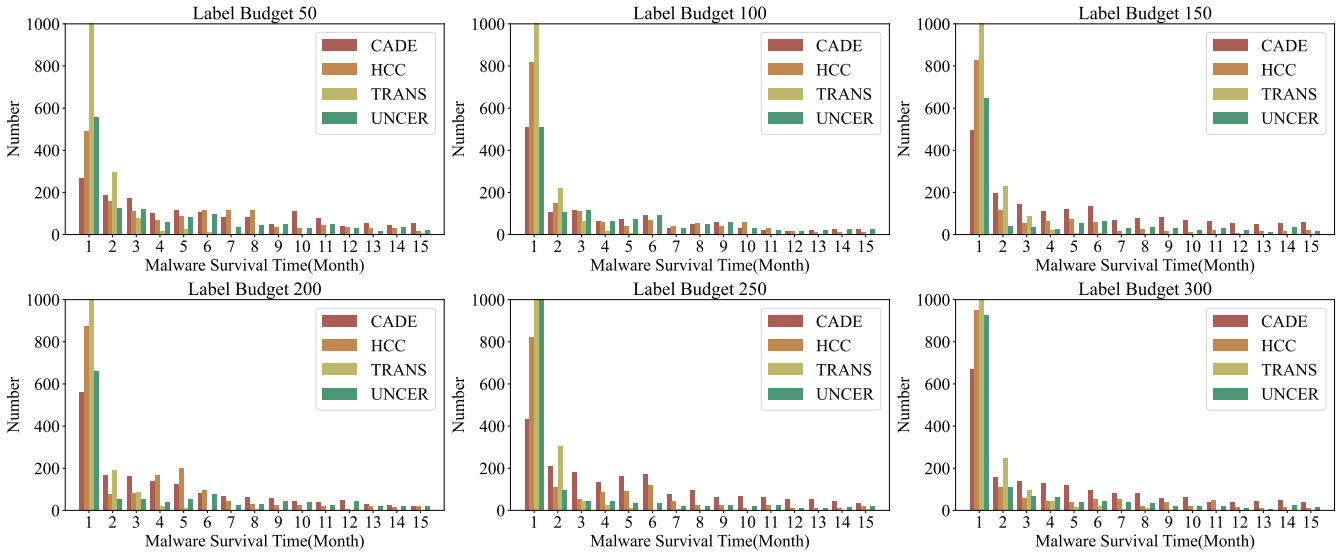

Figure 10: Original misclassification time of attack targets

TABLE 15: 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