CDAD: Concept Drift Adaptation Denial Attack in Android Malware Detection

Abstract

Machine learning-based Android malware detection methods are used to alleviate mobile security threats in the real world. Unfortunately, with the evolution of malware, deployed detection models will soon become outdated, known as concept drift. Concept drift adaptation methods based on active learning are proposed to improve Android malware detection models' performance. The core of concept drift adaptation lies in selecting the most valuable samples during the testing phase and optimizing the model based on these samples. However, we find that we can construct attack samples based on high-value benign samples, controlling sample selection and model updating. Based on the above insights, we propose the concept drift adaptation denial (CDAD) attack to prolong the survival time of new malware. We conduct the first CDAD attack on Android malware detection models constructed by four mainstream concept drift adaptation strategies. We evaluate CDAD attack on an Android malware concept drift dataset containing over 580,000 samples spanning more than 10 years. Experimental results indicate that the latest concept drift adaptation method is vulnerable to our CDAD attack. Attackers can achieve an optimal white-box attack success rate of 84% across four mainstream concept drift methods (reaching 79.75% under the black-box threat model). Furthermore, the impact on the original model's performance (measured by the F1 score) during the attack process is minimal, with an average reduction of less than 2%, demonstrating high stealth in CDAD attack.

1 Introduction

Android operating system has become indispensable to people's lives over the last decade. As of January 2024, the Android operating system ranked first in the global operating system market share, reaching 41.63% [34], with nearly 4.74 billion users worldwide [5]. Unfortunately, mobile devices and applications powered by the Android operating system have been selected as valuable targets by cyber criminals [6]. According to a security analysis by Kaspersky, even the official Google Play Store had over 600 million malware downloads in 2023 [36]. Facing the massive amount of malware generated daily, researchers have proposed an automatic detection method for Android malware based on machine learning [43].

However, deploying Android malware detectors in the real world faces many significant challenges. One of the most critical challenges is that real-world data distribution can change over time, yielding the phenomenon of concept drift [32, 39]. Researchers have demonstrated that Android malware detectors, which performs well on training datasets, experiences a decline in its F1 score from 99% to 76% within approximately 6 months when faced with concept drift [11]. A direct solution is to add new data to the training dataset to ensure that data distribution is consistent with the real-world data distribution. But the number of new Android applications in the real world is overwhelmingly large. Google Play launched 1069 mobile apps every day in 2024 [9]. Therefore, obtaining sample labels for all new data is impossible, which leads to an insufficient quantity of training datasets, ultimately resulting in a decline in the performance of Android malware detection models.

Existing several related landmark papers [7, 11, 14, 41] mainly focus on concept drift adaptation through active learning for Android malware detection. In order to tackle the practical limitations previously mentioned, such as the cost of labeling, the aforementioned research [7,41] has devised diverse evaluation methods aimed at assessing the value of test data. Researchers introduced high-value samples to mitigate model performance degradation caused by concept drift [11, 14]. But most previous research has primarily focused on improving the performance of concept drift adaptation. The vulnerabilities of concept drift adaptation through active learning for Android malware detectors have received little attention.

Previous research either focuses on the training phase (poisoning attacks [40]) or the inference phase (evasion attacks [4, 18, 25, 31]). And we noted that the concept drift adaptation encompasses not only the model inference and training but also the concept drift samples selection, which is the core

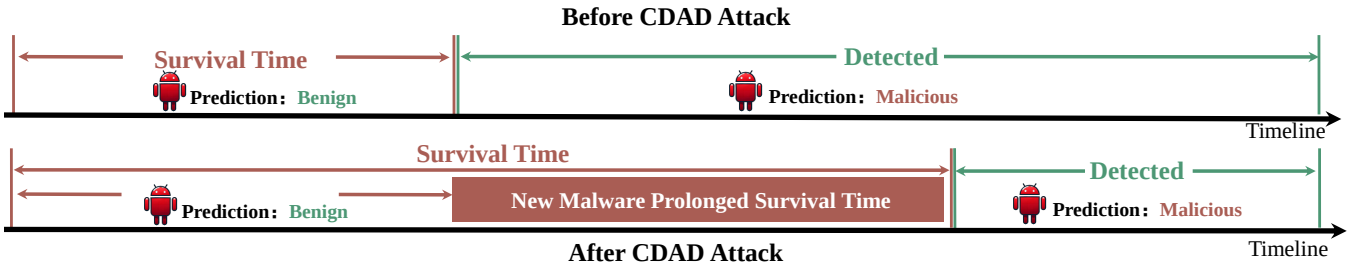

Figure 1: Prolonged Survival Time of New Malware under CDAD Attack

module of concept drift adaptation methods. Therefore, researching the security of concept drift adaptation methods is of utmost importance. The purpose of concept drift adaptation is to shorten the survival time of new malware, so we investigate whether there exists an attack method that can prolong the survival time of new malware(as shown in Figure 1).

In order to study the changes in the survival time of new Android malware under the concept drift adaptation methods, we propose three research questions:
Q1-What is the survival time of new malware samples under latest concept drift adaptation based on active learning [11]?
Q2-Can attackers design an attack method to prolong the survival time of new malware in the face of the optimal concept drift adaptation methods?
Q3-How effective is the attack method, and what factors influence the attack's effectiveness?

We explore the answer to the above question by studying the Android malware detectors with concept drift adaptation capabilities. We choose Android malware because of the availability of public, large-scale, and timestamped datasets(e.g., AndroZoo [1–3, 11, 44]).

To answer Q1, we apply the optimal concept drift adaptation methods [11] against the new Android malware samples. We find that some new malware family samples had a significantly longer survival time than other new malware (§3). For example, the survival time of the tascudap family reached over 2 years. This phenomenon is not isolated. Among the new malware families each month, over 95% of them contain samples with a survival time greater than 0. This implies that there may be certain vulnerabilities in the current concept drift adaptation methods that attackers can exploit. For detailed data on the survival time of new malware, refer to Figure 8 in Appendix B.

To answer Q2, we propose the Concept Drift Adaptation Denial (CDAD) attack (§5) to prolong the survival time of new malware. Using existing real-world Android samples, our proposed method can efficiently generate poison samples with clean labels. Our attack framework comprises three modules: surrogate model training (§5.2.1), malware attack value assessment (§5.2.2), and malware survival time Prolongation (§5.2.3).

To answer Q3, we use the Attack Success Rate (ASR) to measure the attack effectiveness of the CDAD attack, which refers to the proportion of samples whose survival time has been effectively prolonged among all attacked samples. We conducted our evaluation under both white-box and black-box threat models (§4) and analyzed the factors influencing the CDAD attack (§6).

To better demonstrate the vulnerability of mainstream concept drift adaptation methods to CDAD attacks, we conducted experiments on four mainstream Android malware concept drift detection strategies, including Hierarchical Contrastive Classifier (HCC) [11], Transcending (TRANS) [7], high-dimensional outlier distance (CADE) [41], and uncertainty (UNCER) [15]. Our attack evaluation dataset spans 10 years and the total number of samples of our dataset reaches more than 580000. The experimental evaluation results show that our CDAD attack method achieves an average attack success rate of 84%.

Our Contributions. In summary, we mainly make the following contributions:

- We have discovered significant security vulnerabilities in the concept drift adaptation models proposed in recent top security conferences. The survival time of new malware can be effectively prolonged by our CDAD attack. This discovery is helpful for researchers to better understand the persistent attack risks currently faced in the field of concept drift adaptation.
- We propose an automatic sample generation framework for CDAD attack. This framework can efficiently generate attack samples with clean labels, and it does not require any modifications to the malicious samples that need to prolong survival time. This is beneficial for attackers in spreading malware.
- We conducted CDAD attack effectiveness tests on an Android malware concept drift dataset over a period of 10 years, and conducted detailed discussions on the influencing factors of attack effectiveness, attack concealment, and the costs of attackers and defenders. We found that CDAD attack method can achieve an average attack success rate of over 84.02%, with extremely strong concealment (F1 score change less than 0.018 before attack),

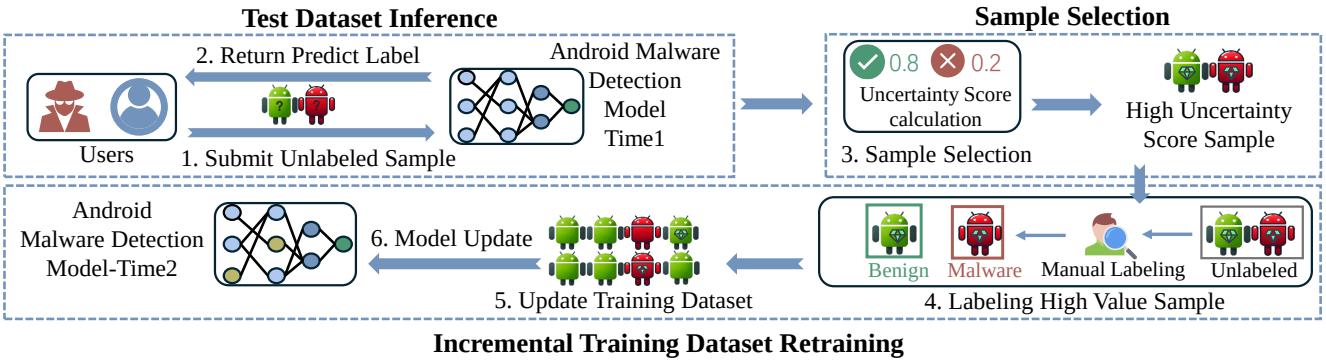

Figure 2: Concept Drift Adaptation Based Active learning

and CDAD attack still has good attack effects (average attack success rate 77.26%) under unfavorable conditions such as unknown target model structure, inability to obtain training data, and limited attack overhead.

2 Background

2.1 Concept Drift Adaptation

Behavior Pattern. Different concept drift adaptation methods also share common behavior patterns (as shown in Figure 2). The classical behavior pattern consists of three sequential execution phases: test dataset inference, sample selection, and incremental training dataset retraining.

Stage1-Test dataset inference: The newly added test data is submitted to the target model for testing. The target model gives a predicted label. In the case of binary classification, the return label value is 0 or 1.

Stage2-Sample selection: The primary goal of sample selection is to find concept drift samples. It can address the issue of how to allocate labeling budgets for new data when the overall labeling budget is limited. Researchers designed different sample selection strategies to find the most helpful samples to improve the model's performance under limited label budget.

Stage3-Incremental training dataset retraining: The model trainer adds selected samples as an incremental update part of the training dataset. After retraining, model performance degradation caused by concept drift can be alleviated.

2.2 Concept Drift Adaptation Strategy

The mainstream concept drift adaptation methods in the field of Android malware detectors include the following four strategies.

1) Model Uncertainty. The core idea of uncertainty measurement [15] is to detect concept drift based on the output layer of the target model. The model will give priority to selecting the samples with high uncertainty of the current model for labeling. A common uncertainty measure for a neural network is to use one minus the max softmax output of the network.

2) Encoder Space Distance. CADE [41] trains an encoder through existing labeled data for learning a compressed representation (dimension reduction) of a given input distribution. Then, the newly obtained test samples can be provided to the encoder to obtain the encoder's spatial features. Finally, the distance function can effectively identify concept drift samples far from the existing dataset for retraining.

3) Credibility and Confidence. Transcending [7] introduced the thery of conformal prediction [37] (credibility and confidence) into the field of concept drift adaptation. Given a new test sample, Transcending first computes the non-conformity score of the sample. Then, it computes credibility as the percentage of samples in the calibration set that have higher non-conformity scores than the test sample. Finally, it computes confidence as one minus the credibility of the opposite label. A lower credibility score or a lower confidence score means the test sample is more likely to have drifted.

4) Hierarchical Contrastive Loss. The method proposed by Chen et al. [11] is currently the best-performing strategy in Android malware concept drift adaptation. The model is separated into two modules. The first module is an encoder and the second module acts as the classifier. In terms of loss function settings, to ensure that the model is robust to concept drift, the training loss of the model is set to the sum of hierarchical contrast loss and classification loss.

2.3 Adversarial Attacks

Adversarial machine learning attacks are being widely studied in multiple fields [8]. Currently, adversarial attacks against Android malware detectors can be roughly divided into two categories: evasion attacks and poisoning attacks.

Evasion Attacks have received extensive attention in the field of Android malware detection [10, 20, 30, 45]. Specifically, the attacker's goal in an evasion attack is to add a small

perturbation to a testing sample to get it misclassified. Such perturbed example is called an adversarial example.

Poisoning Attacks are one of the most dangerous threats to machine learning models [19, 33]. These attacks assume attackers can inject poisoned samples into the training dataset at will. In poisoning attacks, the adversary's goal is to degrade model performance on a validation dataset through some malware modifications to the training dataset. After being trained on the poisoned dataset, the model's performance degrades at test time. According to the different degradation degrees of the target model, poison attacks can be roughly divided into untargeted and targeted poison attacks. The goal of untargeted poisoning attacks is to decline the overall performance of the target model. The goal of targeted poisoning attacks is to force the target model to perform abnormally on a specific input class. Backdoor attacks [19, 22, 33] are a special case of targeted poisoning attacks where the poisoned target models only misclassify samples containing specific triggers.

In summary, existing research has either focused on the model inference security under static conditions (evasion attacks) or the security of the training process of the model (poisoning attacks). However, we found that in addition to the training phase, poisoning attacks are also very likely to occur during the sample selection phase of the concept drift adaptation.

3 Motivation

Our motivation comes from a key real-world observation: the longer the survival time of new Android malware, the more benefits the attacker obtains, such as monetary gain, user privacy, etc. For example, Zimperiumz Lab discovered a new type of Trojan named GriftHorse in 2021 [42], and the infected device would pay the attacker 30 euros a month. This new malware infected over 10 million user devices in over 70 countries and regions in a few months.

Validating the Intuition. In order to understand the survival time of new Android malware, we use the existing optimal concept drift adaptation method (HCC [11]) to quantitatively analyze the survival time of new malware samples.

As shown in Figure 3, some new malware samples still have a long survival time. Among them, families such as 'gomanag' can even survive for more than 5 years (62 months). Under the latest HCC method, the average survival time of new malware families is 4.71 months, while that of new variants is 3.76 months. Therefore, addressing the increasingly prolonged survival time of new malware is crucial.

Research Motivation. Unfortunately, there is a lack of research on how attackers can achieve malware survival time prolongation under concept drift adaptation. Therefore, we investigate the risk of new malware survival time prolongation in the most powerful existing concept drift adaptation approach. Our goal is to reveal the dangers of CDAD attacks and draw the attention of relevant researchers to the model security risks in the concept drift adaptation process of new Android malware.

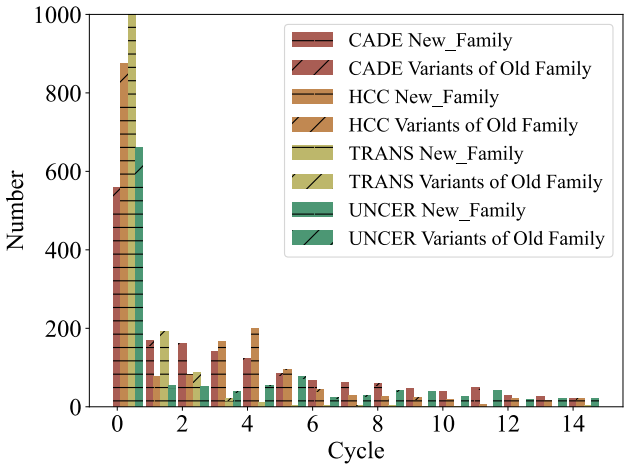

Figure 3: Survival Time of New Malware

4 Threat Model

To ensure the reasonableness of the research hypothesis, we present the threat model based on previous works and real world situations.

In our attack scenario, a capable attacker can carry out attacks based on a white-box threat model. This means that the attacker can access information such as the target model's active learning incremental training dataset, concept drift adaptation strategies, sample feature vectors, and model parameters. This setting follows Kerckhos' principle [26], ensuring that the security of the model does not rely on secrecy. This assumption is realistic since concept drift adaptation methods should be publicly available so that they can be rigorously vetted for security before deployment like cryptanalysis. Additionally, considering that concept drift adaptation may ultimately favor some state-of-the-art methods [12], it will be challenging to maintain concept drift adaptation strategies strict confidentiality.

We further validated the effectiveness of our proposed CDAD attack under the conditions of a black-box threat model. In a black-box threat model [49], the attacker cannot obtain the training dataset or model parameters from the target model. Therefore, the attacker can only rely on a surrogate model for approximate analysis, as demonstrated by previous work [35, 47].

Table 1: Adverisal-Challenge (●:have this characteristic,○:lack of this characteristic)

Attack Method	Attack Continuity	Training out of control	Malware Integrity	Label correctness
Android HIV [10]	○	○	○	●
HRAT [45]	○	○	○	●
Severi et al. [33]	○	○	○	●
Li et al. [19]	○	○	○	●
AdvDroidZero [16]	○	○	○	●
Jigsaw Puzzle [40]	○	○	○	●
Our Work (CDAD)	●	●	●	●

5 Attack Methodology

5.1 Adversary's Challenges

Although attackers{attacker of whole section} have some capabilities and knowledge (mentioned in §4), attacks against concept drift adaptation still face severe challenges. These challenges make our attack significantly different from previous attacks (as shown in Table 1). The detailed challenges faced by attackers are as follows:

Attack Continuity. Previous poisoning attacks [22] typically validate their effectiveness under the condition of a fixed training dataset without considering the scenario where the training dataset is continuously updated. One of the most notable features of active learning is that it continually introduces new data into the training dataset. Therefore, ensuring the continuity of the attack's effectiveness during the model update process is a significant challenge. To the best of our knowledge, this paper is the first to study the Continuity of poisoning attacks on the active learning process for Android malware.

Training Process is out of control. In our threat model, the attacker cannot directly poison the training dataset. They only have sample submission and query access to the latest state of the target model. For example, compared with the attack scheme proposed by Giorgio et al. [33], our attacker challenge has a higher degree of difficulty.

Malware Integrity. Although some attackers have used code updates and new malware samples to counter the detection methods, thereby prolonging the survival time. However, this method requires attackers to pay higher code development costs, and frequent code updates by attackers may alert malware detectors. Therefore, we consider maintaining malware integrity to be a unique challenge in this study.

Label Correctness. The labels of poison samples uploaded by the attacker will not be mislabeled, which is different from the assumption of many advanced model attacks in the image field [46]. The reason is that sample labeling in the active learning process is done manually. This is also a special challenge in active learning attacks.

5.2 Attack Method

Taking the above attack challenge{s} as the prerequisite, we propose Concept Drift Adaptation Denial (CDAD) attack. The overall workflow is shown in figure 4. The first module is to conduct a surrogate model, which is responsible for simulating the target model, providing a basis for subsequent attack steps (§5.2.1). The second module is malware attack value assessment, which is responsible for assessing the attack value of new malware samples and selecting the appropriate attack strategy (§5.2.2). The third module involves the generation of poisoned samples (§5.2.3).

5.2.1 Surrogate Model Training

The attacker builds a surrogate model to obtain the information needed for subsequent attack operations without providing new malware to the target model. The training process of the surrogate model is independent of the training process of the target model. In the initial state{what is initial state}, the parameters of the surrogate model θ_a are consistent with those of the target model θ_d ($\theta_a = \theta_d$). Differences between the surrogate model and the target model emerge from the model retraining stage. Regarding training data acquisition, the owner of the target model is always a large security vendor, so it can effectively collect Android samples D_t and detect concept drift samples D_c. More importantly, the owner of the target model has the ability to conduct reliable sample label analysis on concept drift samples.

However, attackers lack the ability to provide reliable labels for concept drift samples. Attacker collects new data samples D_t from the real world, identifies concept drift samples $\bar{D}_c$, and obtains query results as pseudo labels for concept drift samples $\bar{D}_c$ based on the target model θ_d. It is important to emphasize that due to the openness of the Android platform, the attackers' ability to collect data aligns with that of the target model owner, so they all get Android samples D_t. Refer to Appendix C for further details. {uncomfortable} Additionally, the purpose of the attacker is to approximate the detection ability of the target model θ_d, so he does not need to care about the true label of the concept drift samples

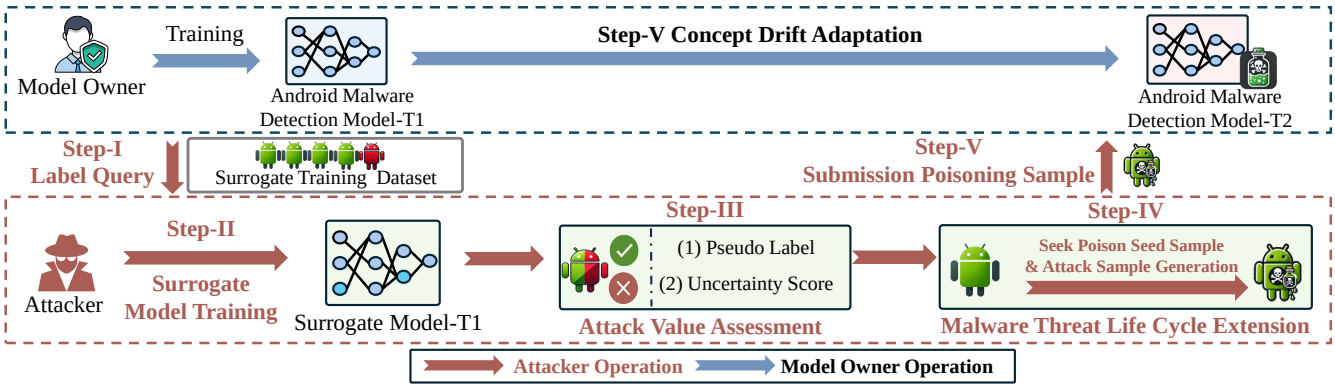

Figure 4: Concept Drift Adaptation Denial (CDAD) Attack

$\bar{D}_c$ but only needs to get the sample prediction result (pseudo label) of the target model θ_d, which greatly reduces the label cost of the attacker. Then, the surrogate model θ_a is retrained based on concept drift samples $\bar{D}_c$ to ensure that its detection performance is always close to the target model.

In addition, our surrogate model construction method differs from the previous shadow model construction methods. Because the target model has the ability to update dynamically, the attacker's surrogate model must also be constantly updated. Considering that our attack scheme has obvious temporal characteristics, we use i to uniformly represent different model update time nodes. We define the interval between the two model update time nodes as a unit model period. In the{a} unit model period, the model parameters of the target model θ_d^i and the surrogate model θ_a^i remain stable. Therefore, after each update of the target model, the attacker queries the target model θ_d^i for the pseudo labels $\hat{Y}_i$ of the latest concept drift samples $\bar{D}_c^i$ and updates the surrogate model θ_a^{i-1} with this{these} samples. By repeating the above process, the attacker will continue to obtain a series of surrogate models $(\theta_a^0, \theta_a^1, ..\theta_a^i...)$ with detection capabilities similar to the series of target models $(\theta_d^0, \theta_d^1, ..\theta_d^i...)$. Each surrogate model object in the model sequence represents the best approximation of the detection ability of the surrogate model to the target model.

Algorithm 1 shows the process of surrogate model training{training process of surrogate model} at time node i. The execution of the algorithm for other time nodes is similar. The concept drift detection function f_c involved in Algorithm 1 corresponds to the four concept drift adaptation methods mentioned in §2.2.

Algorithm 1 Surrogate model training

Input: current time node i, new Android samples D_t^i, target model θ_d^i, active learning label budget LB, loss function L for model training process, training dataset D_T.
Output: Next time node surrogate model sequence θ_a^{i+1}.

Step I: Concept Drift Samples Detection
calculate D_t^i uncertainty scores U_t^i ▷ As defined in 2.2
$U_t^i \leftarrow f_c(D_t^i)$
Step II: Concept Drift Samples Selection
get concept drift samples $\bar{D}_c^i$ (within the label budget LB) based on the uncertainty score U_t^i
$\bar{D}_c^i \leftarrow Select(D_t^i, LB)$
Step III: Pseudo Label Query
query the target model θ_d^i for the pseudo labels $\hat{Y}_i$ of concept drift samples $\bar{D}_c^i$
$\hat{Y}_i \leftarrow Query(D_c^i, LB)$
Step IV: Surrogate Model Retraining
add concept drift sample data $\bar{D}_c^i$ to the surrogate model training dataset D_T
$D_T \leftarrow UpdateDataset(D_T, \bar{D}_c^i)$
retrain the surrogate model θ_a^i based on the training dataset D_T to obtain a new surrogate model θ_a^{i+1}
$\theta_a^{i+1} \leftarrow Training(D_T, L, \theta_a^i)$
return θ_a^{i+1}

5.2.2 Malware Attack Value Assessment

Based on the surrogate model, attackers have the ability to evaluate the attack value of new malware, and then provide differentiated attack strategies. The malware attack value assessment is specifically divided into two sub-modules: false negative attack value assessment{uncomfortable} and uncertainty score attack value assessment.

1) false negative attack value assessment: This part is used to measure the likelihood of new malware surviving in the current model cycle. Specifically, the attacker submits the new malware sample x_m to the surrogate model θ_a^i. If the assessment result of the surrogate model is false negative, it is considered that the new malware sample x_m has a high probability of survival in the current model cycle. If the false negative judgment constraint is not met, the attack value of the current samples x_m is low.

2) uncertainty score attack value assessment: Although

false negative attack value assessment can eliminate samples with no attack value, it cannot prioritize the samples with attack value. Therefore, we propose the uncertainty score attack value assessment.

Specifically, it takes the survival probability in the next model cycle as the basis for judging the attack priority of malware x_m.{specifically} Considering the characteristics of the active learning process, the survival probability of new malware in the next model cycle is usually affected by sample selection caused by the concept drift adaptation strategy. Therefore, when the uncertainty score of new malware x_m is lower than the lowest score u^i_{min} for sample selection in the current model cycle, the new malware will not be subjected to manual analysis. This type of new malware has a higher attack value. To further illustrate the role of the attack value assessment module, we conducted necessity analysis experiments (§6.3).

5.2.3 Malware Survival Time Prolongation

Next, we enter the new malware survival time prolongation module,{Odd} which is the core of the CDAD attack framework. Malware survival time prolongation mainly includes three parts, namely{Odd} poison seed sample selection, poison sample generation, and poison sample data distribution disturbance.

5.2.3.1 Poison Seed Sample Selection We abstract the attack seed sample selection problem into an optimization search problem, searching for benign samples whose uncertainty score ranking is at the top of the label budget in data collected by the attacker. Searching for high-uncertainty samples to use as seeds for generating attack samples ensures the influence on the target model's decision boundary, thereby prolonging the survival time of new malware. It should be noted here that the seed samples of the poisoning attack are benign samples, which provide the basis for clean label attacks.

$$x_k \in (D_T \cup D^i_t) \wedge (y_{x_k} == 0) \tag{1}$$

{what is x_k}We define all the data available to the attacker (including the existing surrogate training dataset D_T and the new Android dataset D^i_t at time node i) as the search space for identifying poisoned attack seed samples (Equation 3). {3? only 3?}

$$X^i_s = \{x_k | f_c(x_k) > u^i_{min}\} \tag{2}$$

The attacker selects samples with uncertainty scores greater than the minimum sample selection uncertainty score u^i_{min} at the current time node i as the seed samples X^i_s for the attack.

5.2.3.2 Poison Sample Generation Poison seed samples have the capability to influence the decision boundary of the target model. However, since the search for seed samples in poisoning attacks is based on optimization strategies, meeting the required capacity for poisoning attack samples is challenging. The range of sample selection in active learning typically involves a set of samples rather than a single sample (Chen et al. (HCC [11]) set label budget as 50, 100, 200, 300). Moreover, the stronger the capability of the target model, the better its sample labeling capacity, resulting in a larger range for sample selection. Too few poisoned samples are difficult to have a big impact on the target model. Therefore, the challenge faced by attackers is how to generate a large number of effective poison samples based on a small number of poison seed samples. It is important to note that the generation of poison samples is called the inverse mapping problem of feature space in problem space in existing research. {in existing research.} That is to say, the attacker needs to write an actual program in the corresponding problem space based on the given features. Therefore, poison sample generation consists of two parts. The first part is a{a} feature space search to confirm the feature space of samples that meet the attack requirements. The second part, feature space inverse mapping, illustrates that the feature space of the poison sample can have a corresponding codable program in the problem space. Since methods for mapping feature space to problem space have already been provided by researchers such as Pierazzi et al. [27], so our attack approach primarily focuses on constructing the feature space for poisoning samples.

Feature Space Search:{search or construct} To maintain the effectiveness of the attack, the attacker adheres to the following three constraints during the feature space construction process.

constraint 1: The poisoned{poisoned or poisoning or poison} samples constructed from the seed samples must have labels consistent with those of the seed samples. This constraint ensures that the generated poisoned samples do not provide the target model with new knowledge about malware samples.

constraint 2: Generated poisoning samples are within the budget of the sample selection in the current model cycle. The uncertainty score of the poisoned sample needs to be higher than the minimum uncertainty in the sample selection budget range sample, thereby ensuring that the poison sample can effectively affect the target model.

constraint 3: The uncertainty score of poisoned samples is higher than that of the malware sample to be protected. This constraint ensures that the generated poisoned samples help the new malware x_m avoid being selected as samples requiring manual analysis. It is also worth noting that constraint 3 only focuses on whether the uncertainty score of the poison sample is higher than that of the sample to be protected and does not consider other samples. Therefore, it is a targeted poisoning attack among poisoning attacks.

Based on the above constraints, we propose a label consistency data perturbation enhancement method based on uncer-

tainty score constraints.{both based on at first} The specific steps are shown in Algorithm 2.

Algorithm 2 New Malware Survival Time Prolongation

Input: time node i, poisoned seed samples X_s^i, surrogate model θ_a^i, minimum sample uncertainty score u_{min}^i within the label budget for the current time node, feature perturbation function f_p ,uncertainty score u_m for new malware x_m

Output: poison samples D_p^i.

Step I: Feature Space Perturbation
while keeping the label stable, perform feature perturbation on samples X_s^i, form perturbed datasets D_s^i.
  for each $k \in len(X_s^i)$ **do**
    $D_{sk}^i = f_p(x_s^k)$ ▷ constraint 1
  end for
  $D_s^i = \left[D_{s0}^i \cup D_{s1}^i \cup .....D_{sp}^i\right] (p = len(X_s^i) - 1)$
Step II: Uncertainty Score Evaluation
perturbed datasets D_s^i are submitted to the surrogate model θ_a^i for testing, and the surrogate model feeds back the uncertainty score.
  for each $j \in len(D_s^i)$ **do**
    $u_j = f_c(p_s^j, \theta_a^i)$, $p_s^j \in D_s^i$
    if $u_j > u_{min}^i$ **then**
      $D_s^i = D_s^i - p_s^j$ ▷ constraint 2
    end if
    if $u_j < u_m$ **then**
      $D_s^i = D_s^i - p_s^j$ ▷ constraint 3
    end if
  end for
  $D_p^i = D_s^i$
  return D_p^i

It should be noted here that the feature perturbation function (f_p) in Algorithm 2-Step I, in order to satisfy the label mentioned in constraint 1, we adopt the perturbation method of positive flipping of malware sample feature space and negative flipping of benign sample feature space. {actually two sentence} This type of disturbance in the feature space corresponds to the problem space in which the malware retains all information and adds some permissions or API call information, or the benign sample removes some permissions or API call information. According to the characteristics of the Android software problem space, it can be seen that the above perturbation method will not change the original sample label, so it meets constraint 1. If the above three constraints are met, then according to the constraints of the poisoned sample, D_p can be used as a poison sample.

At this time, the generated batch poison samples D_p have a more balanced effect in three aspects: ensuring that they enter the label budget range of the current model cycle, label consistency, and generating samples to ensure new malware sample x_m are inevitably not within the sample selection range. Therefore, it can ensure that the generated poison samples effectively change the model decision boundary into a state favorable to the malware sample x_m.

5.2.3.3 Poison sample data distribution disturbance

The above attack method can prolong the survival time of new malware. However, since the poisoned samples are all benign, they may raise the defender's suspicion. To further enhance the stealthiness of the attack, the attacker needs to increase the distribution diversity of the sample selection data used in the poisoning process. But new malware cannot be introduced into the test dataset, as such samples would improve the target model's detection capability. Based on the above requirements, we analyzed the uncertainty scores of samples in the existing training dataset. We found that even in the training datasets of the currently optimal model concept drift adaptation methods, there are still high-uncertainty malware samples.

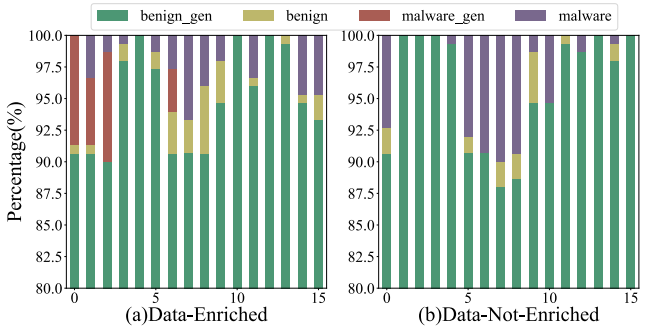

Figure 5: Poison Sample Data Distribution Disturbance

These malicious samples are all from the training data used for the model. Since the current model has classified high-value samples as negative, these samples will not significantly impact the prolongation of the survival time of new malware. At the same time, such samples can enrich the data distribution after sample selection, thereby enhancing the stealthiness of the attack. Therefore, we use these samples to add to the original attack seed samples. Since perturbing the sample data distribution essentially enriches the attack seed samples, it can be seamlessly integrated into the poisoning sample generation algorithm. This simply requires replacing the input seed data, as demonstrated in Algorithm 2. As shown in Figure 5, we conducted statistics on the data distribution of attack samples. Figure 5-a shows the data distribution of concept drift samples after introducing the perturbation module for attack sample data distribution. In contrast, Figure 5-b corresponds to the data distribution without this module. It is evident that the data distribution of concept drift samples on the left is more diverse, encompassing a wider range of malware, which ensures the concealment of attack behaviors.{stealthiness or concealment}

6 Evaluation

6.1 Experimental Setup

6.1.1 Dataset

Android malware concept drift dataset is a kind of special Android malware dataset with time attribute. Compared with the previous classic Android malware datasets, such as Drebin [6], CIC2020 [17, 28], etc., researchers can do more in-depth research on malware evolution characteristics. Currently, there are few datasets of this kind. Researchers often need to collect the data themselves or perform secondary processing on existing non-concept drift datasets.

In this study, we selected two Android malware concept drift datasets (APIGraph [44] and BJTU-AMCD) to validate the effectiveness of our attack methods. Both datasets are characterized by long-term collection and rich family information. We collected and constructed the second dataset. Details on the collection and construction process are in Appendix C. The combined time span of both datasets covers 11 years. The number of samples in the dataset reaches 580,000, with more than 1000 malware families. Detailed information is shown in Table 8.

6.1.2 Target Model Settings

Based on the aforementioned large-scale concept drift datasets, we set our attack objectives. Specifically, this includes the configuration of the model structure and the implementation of adaptation strategies for concept drift. For the target model setup, we based our approach on the current best practices in the field of concept drift (HCC [11]) and enriched it by model mentioned in He et al. [16] and Fang et al. [14]. The specific model setting combination is shown in Table 2. In setting up the concept drift adaptation strategies, we tested not only the current optimal strategy (HCC [11]) but also several key milestone approaches (TRANS [7], CADE [41],UNCER [15]).

Table 2: Combination of Model Structure and CDA-Strategy

Strategy	Structure
HCC [11]	Encoder-Classifier (MLP) [11]
TRANS [7]	Classifier (SVM) [6]
CADE [41]	Encoder-Classifier (MLP) [24]
UNCER [15]	Classifier (ResNet) [14]

6.1.3 Surrogate Model Settings and Evaluation Matrix

Based on the settings of the target model, we provide the corresponding surrogate model settings. The specific settings of the surrogate model are mainly affected by the attacker's capabilities. We classify attackers based on their capabilities into strong attackers and weak attackers. Strong attackers can obtain complete information about the target model and conduct attacks under the white-box threat assumption, as described in §4. Weak attackers, lacking knowledge of the model's structure, parameters, and training dataset, conduct attacks under the black-box threat assumption. A detailed analysis will be provided in §6.2.3. In terms of performance indicators, we use F1 score, False Negative, Accuracy, and other indicators to measure the overall performance of the model performance. Regarding attack effectiveness metrics, we constructed the Attack Success Rate (ASR) based on the survival time of new malware. We define a successful attack as the survival time of new malware being prolonged for no less than one month after the attack.

6.1.4 Attack Objective Settings

To verify the effectiveness of CDAD attack, we selected new malware samples from new family or new varients of old family as the target for testing. Given that the experiment in this paper is a single-sample, multi-cycle testing experiment, the time overhead is relatively large, with a testing time for a single sample ranging from 1 to 4 hours. Therefore, for all samples with attack value, we prioritize testing those with relatively shorter native survival time (less than 6 months). Firstly, the proportion of such samples in the new family is also the highest, reaching more than 90%, as shown in Figure 3. Secondly, these samples have consumed the attacker's development costs but have not provided significant benefits to the attacker, giving them the motivation to prioritize attacking such samples. Based on the aforementioned strategy for selecting attack targets, we extracted 20% of the samples from new malware families for validation of the attack method. In theory, attackers can perform persistent enhancements to the survival time through persistent attacks. However, in order to effectively verify the experiment and define a unified attack success criterion. We define attack success as a 100% amplification of the original survival time.

6.2 Attack Effectiveness

Based on the above settings, we start to evaluate the effectiveness of the CADA attack effect. The evaluation of attack effectiveness is conducted under both white-box and black-box threat models. Additionally, to study the impact of different settings on the effectiveness of the attack scheme, we analyzed various factors influencing the attack.

6.2.1 White Box Attack Effectiveness

Strong attackers have white-box access to model information. We conducted attack tests on 4 models and the corresponding concept drift adaptation strategies. The detailed test data configuration is shown in Table 3.

Table 3: Verification of the effectiveness of concept drift adaptation attack

Concept Drift Strategy	F1	TPR (%)	FPR (%)	TNR (%)	FNR (%)	ASR (%)
HCC (Latest)	$0.90_{-0.02}$	$85.88_{-2.80}$	$0.44_{+0.04}$	$99.56_{-0.04}$	$14.12_{+2.80}$	**89.44**
TRANS	$0.89_{-0.01}$	$85.18_{-1.12}$	$0.53_{\mathbf{-0.01}}$	$99.47_{+0.01}$	$14.82_{+1.12}$	**79.07**
CADE	$0.88_{-0.02}$	$85.57_{-2.62}$	$0.90_{+0.06}$	$99.10_{-0.06}$	$14.43_{+2.62}$	**84.72**
UNCER	$0.88_{+0.01}$	$83.86_{+0.98}$	$0.65_{+0.00}$	$99.35_{+0.00}$	$16.14_{\mathbf{-0.98}}$	**82.86**

First, we observed that the poison samples significantly degrade the target model's ability to detect new malware. Through attack evaluations spanning six years, it can be observed that our proposed attack method can achieve effective attacks, with an average ASR reaching 84%. Specifically, the average ASR can reach 89.44% when targeting the currently optimal method for concept drift adaptation.

At the same time, we also noted the attack's impact on the detection performance of other malware. The purpose of targeted poisoning attacks carried out by attackers is to prolong the survival time of new malware. Therefore, while ensuring the success of the attack, the attacker prefers to minimize the overall impact on the model's detection performance. To quantify the target model's detection capability on non-target malware before and after the attack, we selected F1 as the performance metric for the model. If the change in the model's F1 metric before and after the attack is not significant, it is considered that the attacker has achieved the concealment of the attack operation while ensuring the effectiveness of the attack. It can be observed that the F1 performance indicator varies within 0.018. Even under the UNCER setting, the F1 performance has a slight increase compared with active learning process without attack, indicating the concealment of our attack. From the user's perspective, the average TNR of the model under CADA attacks reaches 99%, which will not cause any disturbance to users due to false alarms.

Additionally, it is important to note that the ASR metrics indicate the attack success rate under the current dataset test. In real-world scenarios, the threat posed by attackers would be even more severe. On one hand, samples from the same family as the successful poison samples can also benefit from the attack. We analyzed 57 successful attack samples from new families under the pseudo-loss setting (spanning 6 years) and found that when one sample in a new family is successfully attacked, 94.73% of these families have other samples that can prolong their survival time, even without being attacked. On the other hand, attackers can create variants based on the successfully attacked targets, resulting in a large number of false negative samples.

6.2.2 Attack Influencing Factors

So far, we have shown that our proposed CDAD attack is effective against a large amount of new malware samples. However, we currently lack analysis on the impact of differences in attack scheme settings on attack effects. Understanding the differences in attack settings not only helps us more fully understand the vulnerabilities of existing concept drift adaptation schemes but also helps model defenders gain a deeper understanding of attack schemes and thereby propose effective defense methods in the future. The influencing factors during the attack process mainly include three parts, namely the label budget, the proportion of label budget occupied by attackers, and the different feature extraction methods

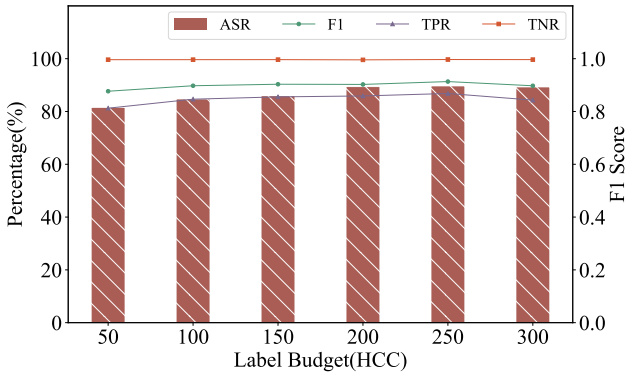

Figure 6: Attack Effectiveness under Different Label budget

1) Label budget: Label budget is an important parameter in the concept drift adaptation process, because it corresponds to the extremely valuable manual labeling cost of security companies in the real world. In order to fully explore the impact of label budget on concept drift adaptation attack, we set up multiple sets of comparative experiments for analysis. The experimental data refers to Figure 6. It can be seen that no matter how much the label budget is, the attack is effective. And we noticed that as the label budget increased, the attack effect also improved synchronously. The setting of a label budget of 300 increased by 7.77% compared to the setting of a label budget of 50. The reason is that as the label budget expands, the impact of the attack samples on the sample selection of the target model will also expand synchronously, which will further enhance their influence on the detection capability of the target model. This leads us to an important conclusion. The higher the cost of manual label analysis paid by the target model, the more favorable it is for attackers to launch attacks.

2) Label budget occupied by attackers: The proportion

of poisoning samples within the label budget represents the intensity of the attack. The higher the proportion of poisoning samples within the label budget, the greater the attacker's attack cost. To effectively illustrate the impact of label budget proportion on attack effectiveness, we set the label budget proportions to 100%, 70%, 50%. We then evaluated how different label budget proportions affect the attack outcome. As shown in Table 4, different label budget proportions have varying impacts on ASR. Although the average ASR of multiple sets of attack tests can still reach 85.88%. However, we can observe a clear trend in ASR: As the proportion of label budget decreases, ASR gradually declines. Specifically, the settings of 70% and 50% proportions result in a decrease of 2.77% and 7.92% in ASR, respectively.

Table 4: proportion of label budget occupied by the label budget

Proportion	F1	FPR (%)	FNR (%)	ASR (%)
100	0.90	0.44	14.12	**89.44**
70	0.90	0.43	14.22	**86.67**
50	0.90	0.46	14.21	**81.52**

3) Different Feature Extraction Methods: Considering that Android malware detectors, in practice, may adopt different feature extraction methods, we extracted static features [6], such as permissions, from our self-constructed dataset to conduct attack experiments on heterogeneous features. The attack results are shown in Table 5. It can be seen that our proposed CDAD attack can achieve effective attacks (with an ASR of over 90%) against different feature extraction methods.

Table 5: Feature Heterogeneity

Feature	F1	FNR (%)	ASR (%)
API [44]	$0.90_{-0.02}$	$14.12_{+1.12}$	**89.44**
Drebin [6]	$0.67_{+0.01}$	$44.57_{+1.39}$	**91.96**

6.2.3 Black Box Attack Effectiveness

To demonstrate the effectiveness of conducting CDAD attacks under the black-box threat model, we have set up the role of a weak attacker. Specifically, compared to the strong attacker setting, we made the following adjustments for the weak attacker. In terms of mastery of target model information, weak attackers cannot access the parameters and training dataset of the target model. Since complex models represent greater computational overhead, we have weakened the model settings for the attacker. Considering that the current optimal concept drift adaptation methods mainly consist of an encoder and a classifier, we have provided four sets of comparative settings based on the target model (as shown in Table 6).

Table 6: Attack Effectiveness Under Model Heterogeneity

Model	F1	ASR (%)	R-ASR (%)
Enc-Cla	0.89	**77.27**	**82.95**
Enc↓-Cla	0.87	**72.31**	**59.09**
Enc-Cla↓	0.83	**82.22**	**44.31**
Enc↓-Cla↓	0.86	**87.18**	**39.77**

Previous research has indicated that clean-label attacks suffer from end-to-end performance degradation [38], as model updates can lead to a deterioration in attack effectiveness. However, the CDAD attack alleviates the issue of diminished attack effectiveness under end-to-end conditions. The experimental result data (Table 6) shows that the average ASR under various black-box settings can reach 80.57%. The reason is that although the attacker may not have complete knowledge of the target model in an end-to-end setting, the CDAD attack can effectively influence the data that the target model relies on for updates. Therefore, CDAD indirectly impacts the target model's updates, enhancing the attack's effectiveness in an end-to-end setting. This demonstrates that, under the assumption of a black-box model, our attack method still poses a significant security threat to the currently optimal concept drift adaptation methods.

Moreover, we noticed that the reduction in ASR caused by weakening the encoder is more pronounced than that caused by weakening the classifier, with a difference of nearly 10%. This indicates that the leakage of encoder information poses a greater threat to the target model. The results of our experimental analysis also echo the current best concept drift methods that rely on the encoder to learn the similarities among malware families.

$$\textbf{R-ASR} = \frac{\textbf{NSAS(Weak\_Model\_Settings)}}{\textbf{NSAS(Equal-Enc\&Cla)}} \tag{3}$$

Additionally, we have also observed an interesting phenomenon. The ASR under the synchronized weakening of both the encoder and classifier are the highest among all settings, even surpassing the control group under model alignment by approximately 10%. To investigate the reasons behind this phenomenon, we analyzed the selection of attack targets under different settings. We found that while the synchronized weakening setting exhibits an advantage in terms of the ASR metric, it demonstrates a clear disadvantage in the number of attack targets. The absolute number of attacks in the synchronized weakening setting only accounts for 55% of the attacks in the control setting. Therefore, we conclude that due to the difference in attack value assessment capabilities resulting from model misalignment, the evaluation of attack effectiveness under the black-box assumption should consider the ASR and the number of attack samples. We define the **Number of Successful Attack Samples (NSAS)**.

Subsequently, we define the **Relative Attack Success Rate (R-ASR)** as the ratio of the number of successful attack samples in the attacker's setup group (numerator) to the number of samples in the control group (denominator). Using the R-ASR metric, we can observe that as the model's capability weakens, the attack effectiveness gradually decreases. Specifically, the weakest setup exhibits a 43.18% reduction in relative attack success rate compared to the control setup.

6.3 Attack Value Assessment necessity analysis

Analysis of the attack value of new malware samples provides support for subsequent survival time extension modules. In order to fully illustrate the role of this part, we conducted relevant ablation experimental analysis. Specifically, attackers skip the attack value assessment phase and directly move to the seed sample selection and attack sample generation phases. This means that attackers operate on all new malware, significantly increasing the cost of attacks. In an ablation study involving 3 different concept drift strategies and label budgets (200), the absence of the attack value assessment module led to an average decrease of 13.78% in attack success rates. Notably, the attack success rate of the UNCER strategy group showed the most significant change, improved by 21.36%. A comparison of attack effectiveness results reveals that the absence of attack value assessment leads to reduced attack effectiveness. The average performance change is illustrated in Table 7. This demonstrates that the attack value assessment module is an essential part of CDAD attacks.

Table 7: Attack Value Assessment necessity analysis

Strategy	Ablation-ASR(%)	ASR(%)
HCC	81.73	**89.44**$_{+7.71}$
CADE	72.43	**84.72**$_{+12.29}$
UNCER	61.50	**82.86**$_{+21.36}$

6.4 Attack Explainability

This section further analyzes and explains why the attack is effective from the perspective of the model decision boundary. Taking the 'execdownload' family that emerged in the real world in June 2015 as an example, we illustrate the effectiveness of CDAD attacks. As shown in Figure 7, in the absence of attacks, we can observe that the decision boundary of the model gradually changes during concept drift adaptation. New malware samples, which were originally far from the decision boundary, gradually move closer to it and are ultimately detected as malware. However, with the introduction of CDAD attacks, we can observe that the model's decision boundary enters an abnormal state of updating. New malware family samples are always far away from the decision boundary, which leads to the continuous expansion of the survival time. The reason for the above phenomenon is that the change of the decision boundary depends on high-uncertainty samples. Although poison samples have high uncertainty and include both benign and malware samples, the contribution of poison samples to the update of the decision boundary is very small. Ultimately, this leads to difficulties in effectively updating the decision boundary of the target model, resulting in an prolonged survival time for new malware.

7 Discussion and Future Work

In this study, we introduce an efficient framework for Concept Drift Adaptation Denial (CDAD) attacks. In the following, we discuss some limitations of our attack method and outline potential future research directions.

Limitation of CDAD attack:In the process of CDAD attack, it is necessary to rely on poisoning attack seed samples with high uncertainty scores. However, in some cases, such samples may not appear in the test dataset of the current month. To enable CDAD to adapt to such scenarios, we have adopted strategies to relax the uncertainty score constraints or employ a freeze attack strategy for attack operations. This approach can effectively help attackers solve the problem of inability to obtain attack seed samples. Nevertheless, we acknowledge that this approach may potentially reduce the success rate or stealthiness of the attack. In future research, we plan to tackle these limitation.

Defense Strategy: As we know, the current adaptation methods for concept drift based on active learning lack corresponding research on defense strategies. Only Lin et al. [23] have conducted research on defense strategies against poisoning attacks in active learning. However, the current method assumes that the attacker has the ability to maliciously mislabel sample label, while optimal concept drift detection method relies on manual labeling. We consider CDAD attack mitigation strategies from the perspective of concept drift sample detection. We believe that an attack sample screening module should be further introduced for concept drift adaptation. The core difficulty lies in how to ensure the improvement of concept drift adaptation performance while reducing the success rate of poisoning attacks. Especially when considering that a large number of attack samples are based on clean-label settings, it is often difficult to distinguish them from other benign samples.

8 Related Work

This work is broadly related to works on the survival time of Android malware, concept drift adaptation and poisoning attacks.

Malware Survival Time. Research on the survival time of malware primarily focuses on the dynamic changes of mal-

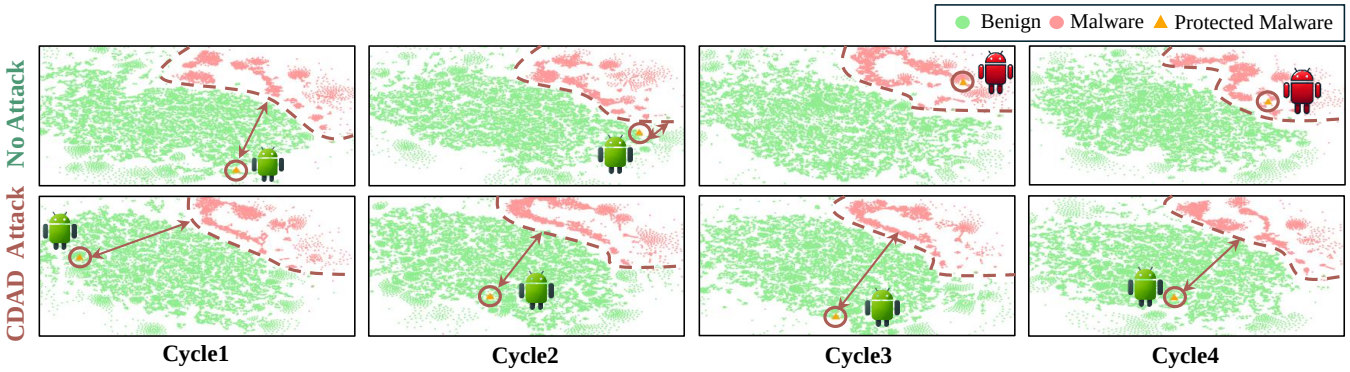

Figure 7: Attack Explainability

ware detection results and their influencing factors over time. Shuofei Zhu et al. [48] have been the first to observe fluctuations in the survival time of malware based on large-scale data collection spanning a year. However, this research primarily focuses on the detection quality of malware detection engines and how to interpret the detection results of different engines. This work attributes the fluctuations in the survival time of malware to the detection quality of the detection engines. Still, there is a lack of research on the reasons behind the differences in detection quality among these engines. Inspired by the large-scale data analysis and statistics [48], our research focuses on the influencing factors of the survival time of malware.

Attack for Android Malware Detectors. The attack methods targeting Android malware detectors are primarily categorized into two types. 1) Evasion Attack. Attackers can obtain false-negative detection results for new malware by performing operations such as repackaging and manipulating control flow graphs [10, 21]. The currently optimal method, proposed by Ping He et al. [16], can achieve effective attacks under a zero-knowledge setting. 2) Backdoor Attack for AMD. Apart from modifying malware, attackers have also proposed backdoor attack methods that exploit security vulnerabilities in the training phase of malware detection models [13, 19, 33]. The current mainstream method is to carry out backdoor attacks by adding triggers to new malware [40], thereby prolonging its survival time. However, both of the aforementioned methods require modifications to the new malware, which not only increases the attack cost but also hinders its widespread dissemination. Therefore, we propose CDAD, a targeted poisoning attack method that does not require any modifications to the malware samples. It effectively prolongs the survival time of new malware while significantly reducing attack costs.

9 Conclusion

In this paper, we present the first poison attack in the concept drift adapatation strategies for Android malware. We design an automatic attack sample generation framework called CDAD to efficiently perform concept drift adaptation denial attacks. CDAD can effectively prolong the survival time of new malware without requiring any modifications to the protected malware samples, while also exhibiting extremely high attack concealment. Our evaluation of a decade-long real-world dataset shows that current concept drift adaptation strategies are ineffective against CDAD. Our work provides researchers with a deeper understanding of the ongoing attack risks in concept drift adaptation.

References

[1] ApkTool. https://apktool.org/.

[2] VirusTotal. https://www.virustotal.com/.

[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 468–471, New York, NY, USA, 2016. ACM.

[4] Hyrum S Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth. Learning to evade static pe machine learning malware models via reinforcement learning. *arXiv preprint arXiv:1801.08917*, 2018.

[5] appsealing. Antivirus for android. https://www.appsealing.com/thank-you-for-showing-interest-in-the-threat-landscape-report-2024/#whitepaper, 2024.

[6] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[7] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 805–823. IEEE, 2022.

[8] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156, 2018.

[9] BIGOH. Top google play store statistics 2024 – exploring the key insights. https://bigohtech.com/google-play-store-statistics/, 2024.

[10] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. Android hiv: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security*, 15:987–1001, 2019.

[11] Yizheng Chen, Zhoujie Ding, and David Wagner. Continuous learning for android malware detection. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1127–1144, Anaheim, CA, August 2023. USENIX Association.

[12] Tianshuo Cong, Xinlei He, Yun Shen, and Yang Zhang. Test-time poisoning attacks against test-time adaptation models. *arXiv preprint arXiv:2308.08505*, 2023.

[13] Mario D'Onghia, Federico Di Cesare, Luigi Gallo, Michele Carminati, Mario Polino, and Stefano Zanero. Lookin'out my backdoor! investigating backdooring attacks against dl-driven malware detectors. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 209–220, 2023.

[14] Wenbo Fang, Junjiang He, Wenshan Li, Xiaolong Lan, Yang Chen, Tao Li, Jiwu Huang, and Linlin Zhang. Comprehensive android malware detection based on federated learning architecture. *IEEE Transactions on Information Forensics and Security*, 2023.

[15] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

[16] Ping He, Yifan Xia, Xuhong Zhang, and Shouling 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*, pages 90–104, 2023.

[17] David Sean Keyes, Beiqi Li, Gurdip Kaur, Arash Habibi Lashkari, Francois Gagnon, and Frédéric Massicotte. Entroplyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics. In *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, pages 1–12. IEEE, 2021.

[18] Yunus Kucuk and Guanhua Yan. Deceiving portable executable malware classifiers into targeted misclassification with practical adversarial examples. In *Proceedings of the tenth ACM conference on data and application security and privacy*, pages 341–352, 2020.

[19] Chaoran Li, Xiao Chen, Derui Wang, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. Backdoor attack on machine learning based android malware detectors. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3357–3370, 2021.

[20] Deqiang Li and Qianmu Li. Adversarial deep ensemble: Evasion attacks and defenses for malware detection. *IEEE Transactions on Information Forensics and Security*, 15:3886–3900, 2020.

[21] Heng Li, ShiYao Zhou, Wei Yuan, Jiahuan Li, and Henry Leung. Adversarial-example attacks toward android malware detection system. *IEEE Systems Journal*, 14(1):653–656, 2019.

[22] Xiaoguang Li, Ninghui Li, Wenhai Sun, Neil Zhenqiang Gong, and Hui Li. Fine-grained poisoning attack to local differential privacy protocols for mean and variance estimation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1739–1756, 2023.

[23] Jing Lin, Ryan Luley, and Kaiqi Xiong. Active learning under malicious mislabeling and poisoning attacks. In *2021 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[24] Lucky Onwuzurike, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini. Mamadroid: Detecting android malware by building markov chains of behavioral models (extended version). *ACM Transactions on Privacy and Security (TOPS)*, 22(2):1–34, 2019.

[25] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[26] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[27] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[28] Abir Rahali, Arash Habibi Lashkari, Gurdip Kaur, Laya Taheri, Francois Gagnon, and Frédéric Massicotte. Didroid: Android malware classification and characterization using deep image learning. In *Proceedings of the 2020 10th International Conference on Communication and Network Security*, pages 70–82, 2020.

[29] Abir Rahali, Arash Habibi Lashkari, Gurdip Kaur, Laya Taheri, Francois Gagnon, and Frédéric Massicotte. Didroid: Android malware classification and characterization using deep image learning. In *Proceedings of the 2020 10th International Conference on Communication and Network Security*, pages 70–82, 2020.

[30] Hemant Rathore, Sanjay K Sahay, Piyush Nikam, and Mohit Sewak. Robust android malware detection system against adversarial attacks using q-learning. *Information Systems Frontiers*, 23:867–882, 2021.

[31] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic black-box end-to-end attack against state of the art api call based malware classifiers. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, pages 490–510. Springer, 2018.

[32] Jeffrey C Schlimmer and Richard H Granger. Incremental learning from noisy data. *Machine learning*, 1:317–354, 1986.

[33] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *30th USENIX security symposium (USENIX security 21)*, pages 1487–1504, 2021.

[34] Statcounter. Operating system market share worldwide. https://gs.statcounter.com/os-market-share, 2024.

[35] Minxue Tang, Anna Dai, Louis DiValentin, Aolin Ding, Amin Hass, Neil Zhenqiang Gong, and Yiran Chen. Modelguard: Information-theoretic defense against model extraction attacks. In *33rd USENIX Security Symposium (Security 2024)*, 2024.

[36] Alanna Titterington. Google play malware clocks up more than 600 million downloads in 2023. https://www.kaspersky.co.uk/blog/malware-in-google-play-2023/26904/, 2023.

[37] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[38] Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. Threats to training: A survey of poisoning attacks and defenses on machine learning systems. *ACM Computing Surveys*, 55(7):1–36, 2022.

[39] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101, 1996.

[40] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 719–736. IEEE, 2023.

[41] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2327–2344, 2021.

[42] Aazim Yaswant. Grifthorse android trojan steals millions from over 10 million victims globally. https://www.zimperium.com/blog/grifthorse-android-trojan-steals-millions-from-over-10-million-victims-globally/, 2021.

[43] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, 50(3):1–40, 2017.

[44] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min 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*, pages 757–770, 2020.

[45] Kaifa Zhao, Hao Zhou, Yulin Zhu, Xian Zhan, Kai Zhou, Jianfeng Li, Le Yu, Wei Yuan, and Xiapu Luo. Structural attack against graph based android malware

detection. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3218–3235, 2021.

[46] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International conference on machine learning*, pages 7614–7623. PMLR, 2019.

[47] Jianing Zhu, Xiawei Guo, Jiangchao Yao, Chao Du, Li He, Shuo Yuan, Tongliang Liu, Liang Wang, and Bo Han. Exploring model dynamics for accumulative poisoning discovery. In *International Conference on Machine Learning*, pages 42983–43004. PMLR, 2023.

[48] Shuofei Zhu, Jianjun Shi, Limin Yang, Boqin Qin, Ziyi Zhang, Linhai Song, and Gang Wang. Measuring and modeling the label dynamics of online {Anti-Malware} engines. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2361–2378, 2020.

[49] Yuanxin Zhuang, Chuan Shi, Mengmei Zhang, Jinghui Chen, Lingjuan Lyu, Pan Zhou, and Lichao Sun. Unveiling the secrets without data: Can graph neural networks be exploited through data-free model extraction attacks?

A Notation

Symbol	Description
t_i	Time interval between the $(i-1,i)$-th model updates
θ_d^i	Target model after the i-th update
θ_a^i	Surrogate model after the i-th update
D_c^i	Concept drift samples at t_i
D_t^i	Testing samples at t_i
$\hat{Y}_i$	Pseudo labels of concept drift samples D_{ci}^i
f_c	Concept drift detection function
LB	Active learning label budget
L	Active learning loss function
D_T	Surrogate model Training dataset
u_{min}^i	Minimum score in the i-th model cycle
x_m	Protected malware sample
$\hat{y}_m$	Predicted label of protected malware sample
v_m	Attack value of malware
u_m	Uncertainty score of protected malware sample
X_s	Selected Poison Seed Sample
τ	Minimum score of high-uncertainty samples
D_p^i	Generated batch poison samples
M	Original feature space dimension of Samples
D_s	Perturbed dataset for Samples from X_s

B Original survival time of all budget

We have analyzed the survival time of new malware under different label budget settings and various concept drift adaptation strategies, as shown in Figure 8. Most new malware can survive for 1-5 months under concept drift adaptation strategies, while a small portion can survive for more than 5 months. It can be seen that new malware has a significantly longer survival time compared to other types of malware.(Due to the concentration of the sample size in the TRANS strategy when plotting, we truncated the TRANS samples while maintaining the relationship of sample sizes to facilitate presentation.)

C Collection of Concept Drift Datasets

Table 8: Evaluation Dataset

Year	Malware	Benign	Malware Family
2012	3061	27472	104
2013	4854	43714	172
2014	5809	52676	175
2015	5508	51944	193
2016	5324	50712	199
2017	2465	24847	147
2018	3783	38146	128
Total	**30804**	**289511**	**1118**
2017	2108	18972	192
2018	4625	41625	363
2019	8612	77508	354
2020	3512	31608	283
2021	4903	44127	256
2022	2814	25326	136
Total	**26574**	**239166**	**1584**

We obtain continuously updated lists of Android applications (latest.csv) from the Androzoo [3] official website. The latest data in this list is continuously updated until 2024, but we have found that the amount of data after 2023 is insufficient to support model training convergence. Therefore, we chose to construct the dataset using data up to the year 2022. Afterwards, we classified the APKs into different time windows based on the "vt_scan_date" field in the CSV file, and categorized them as benign or malware based on the "vt_detection" field. For malware, we collected their family category information from the VirusTotal [2] website to form the raw data of our own Android malware dataset. In the feature extraction phase, we first use the apkTool [1] tool to decompile the APK package. Afterwards, we extract the APK features based on the static analysis method proposed

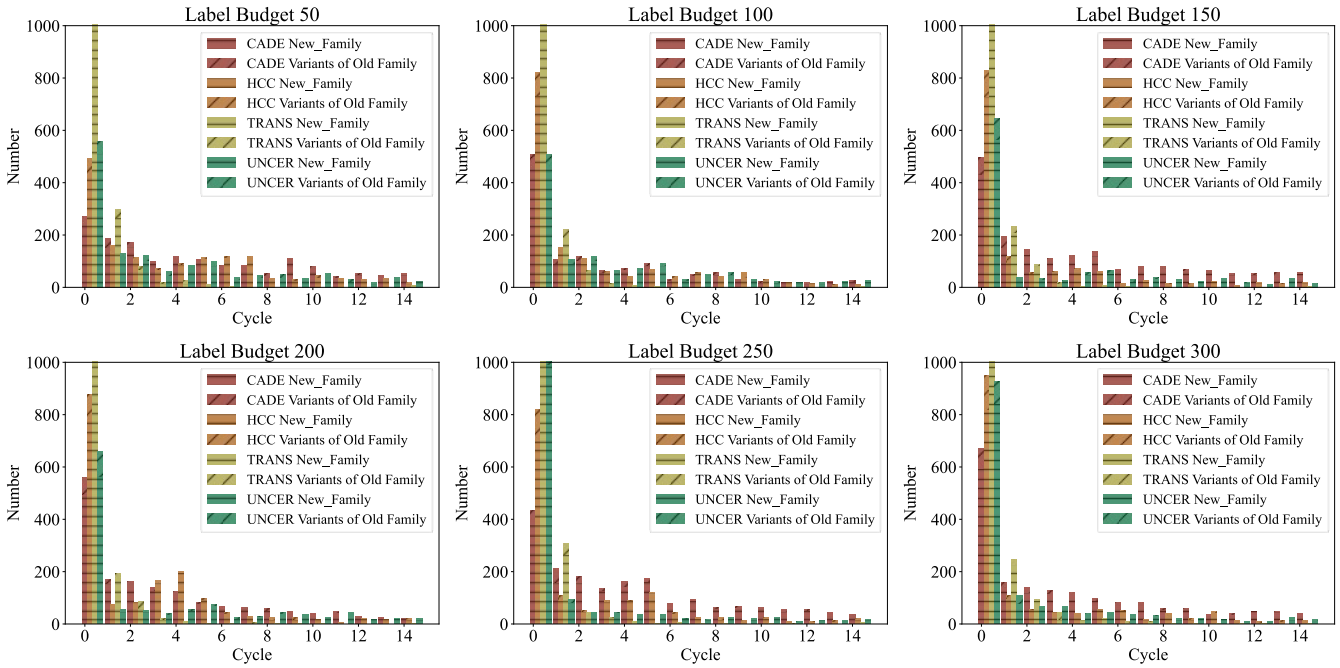

Figure 8: Original survival time of all budget

by Rahali et al. [29], resulting in the final Android malware concept drift dataset. For detailed feature information, please refer to Table 9.

Table 9: Feature information

Features	**Number**	**Example**
Permissions	887	internet, vibrate, bluetooth...
Services	4428	sync job, channel, process...
Actions	1246	pause, reboot, search...
Categories	84	launcher, game, proxy stub...
Total number: 6645		

D Concept Drift Adaptation Baseline

We conducted baseline tests on the performance of current mainstream concept drift adaptation methods on the APIGraph [44] and BJTU datasets, and the test results are shown in the Figure 9. We run all experiments on a Win11 with 96GB memory, 1 Intel(R) Core(TM)i7-14700K 3.4GHz CPU and one NVIDIA GeForce RTX 4080 SUPER (16GB). The model hyper parameters and network structure information are shown in Table 10. To ensure the convergence of the model, we configured the hyperparameters as follows: We utilized the SGD optimizer for HCC, TRANS, and UNC while employing the ADAM optimizer for CADE. The learning rates were set based on the different active learning methods, with HCC, TRANS, and UNC having a learning rate of 0.003 and CADE having a learning rate 0.0001. The learning rate decay was also set to 0.05 for HCC, 0.95 for TRANS and UNC, and 1 for CADE. Regarding Decay Epochs, HCC, CADE, and TRANS were set to start decaying from the 10th epoch, with decay occurring every 10 epochs and the final decay at the 50th epoch. UNC, on the other hand, was set to start decaying from the 30th epoch, with decay occurring every 30 epochs and final decay also at the 50th epoch. The batch sizes were configured as 32 for CADE, 512 for TRANS, and 1024 for both HCC and UNC. For the loss functions, TRANS used the Hierarchical Distance loss function, CADE used the Triplet Mean Squared Error loss function, and HCC and UNC employed the Hierarchical Distance Cross-Entropy loss function. The learning rate scheduling strategies included the Step Learning Rate Scheduler for HCC, TRANS, and UNC, while CADE used the CosineAnnealingLR. The total number of training epochs was set to 50 for all four methods.

The performance indicators of the concept drift adaptation method are shown in Figure 9. It can be observed that the higher the label budget setting is, the better the performance of the concept drift adaptation method becomes. Moreover, the latest concept drift adaptation method (HCC [11]) demonstrates a significant advantage over the existing methods [7, 15, 41].

In this study, we tested many different types of current mainstream excellent Android malware detection models [44] [14] [11], and the obtained concept drift detection data is shown in Figure 9. As can be seen from the

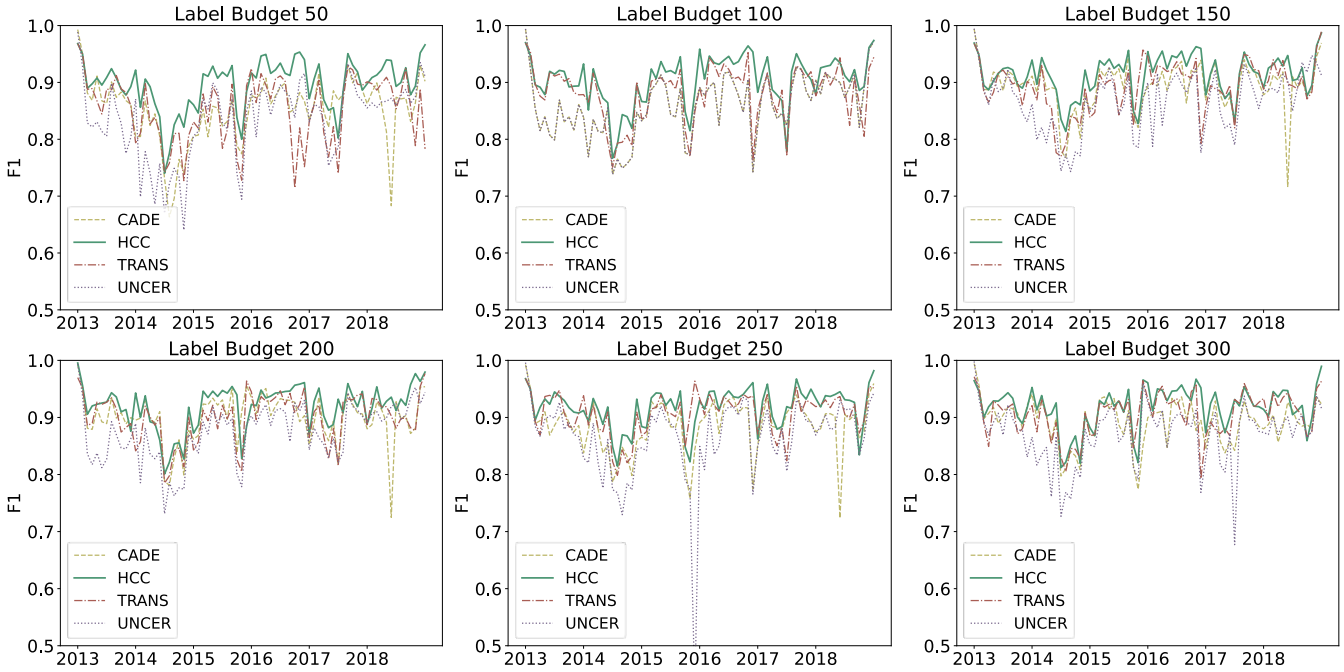

Figure 9: Concept Drift Adaptation Baseline

figure, after 6 years, the detector's F1 performance index averaged 72.92% per month, a decrease of 26.37% compared to the initial month, and the lowest point dropped to 31.87% in November 2015, with an FNR as high as 77.66%.

Table 10: Parameter setting of active learning method

Parameter	**Method**	
	HCC	**CADE**
Optimizer	SGD	ADAM
LR	0.003	0.0001
Batch size	1024	32
Loss	hi-dist-xent	triplet-mse
LR decay	0.05	1
Decay epochs	10,500,10	10,500,10
Scheduler	step	cosine
learning epochs	50	50
Parameter	**TRANS**	**UNC**
Optimizer	SGD	SGD
LR	0.003	0.003
Batch size	512	1024
Loss	hi-dist	hi-dist-xent
LR decay	0.95	0.95
Decay epochs	10,500,10	30,1000,30
Scheduler	step	step
learning epochs	50	50

E Time Cost of CDAD Attack

To fully demonstrate the rationality of attack operations in the problem space, we tested the time cost of repackaging and obfuscation operations of various APK programs in the real world. Our purpose is to demonstrate that attackers can quickly map attack samples from the feature space to the problem space. We selected APKs of different types and sizes,and tested their corresponding repackaging and obfuscation time, as shown in Table 12.

Table 12: 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

Based on the time-based test data, we can observe that the average attack time overhead for a single sample in the problem space is less than 5 minutes. Since the concept drift adaptation model is typically updated on a monthly basis, attackers have ample time to carry out their attacks.

After fully confirming the effectiveness of the attack

Table 11: Attack Time Cost

Stage	Concept Drift Strategy and Active Learning Label Budget											
	CADE						HCC					
	50	100	150	200	250	300	50	100	150	200	250	300
Stage	**TRANS**						**UNC**					
	50	100	150	200	250	300	50	100	150	200	250	300
Seed Selection	8.63	13.2	14.93	16.29	18.15	17.77	0.08	0.1	0.06	0.08	0.08	0.12
Sample Generation	2.52	2.63	2.82	3.05	3.17	2.57	0.1	0.03	0.03	0.03	0.1	0.2
Model Update	1.78	1.7	1.77	1.84	1.9	1.7	3.7	2.88	1.15	5.68	2.32	3.78

method, we conducted tests on the attack time cost. We evaluated data from 2013 to 2018 and found that the current optimal concept drift adaptation method has an average feature space attack time cost of 5 minutes and 49 seconds. Considering the differences in the sizes of malware packages, the time cost for problem-space attacks varies significantly. Therefore, we selected various types of software, including e-commerce, gaming, and social media, to test the time cost of problem-space attack operations. The experimental results show that the average time cost for a single-sample problem-space attack operation is 6 minutes and 8 seconds. In summary, the total time cost of the entire attack process is significantly lower than the model update frequency of mainstream concept drift adaptation methods, which update models on a monthly basis. For specific configuration parameters of the test environment for attack time cost, refer to the Appendix D.

F Attacker and defender cost analysis

The defender's cost mainly comes from the manual labeling cost of new concept drift within the sample selection range during active learning. The exact cost depends on the label budget range. The attacker's cost structure is more complicated, including manual labeling costs, attack seed sample search costs, and sample construction costs. Since the attacker's surrogate model training phase uses pseudo labels, there is no manual labeling cost in the active learning process. The main labeling cost is used for attack seed sample search. According to actual test analysis of a large number of poison samples, the average search range is xx samples per month, which is much smaller than the labeling cost of the defender's active learning process.

Finally, the attacker needs to pay a certain cost to construct the poison sample. This part involves how to construct the corresponding problem space software program object after the feature space is determined. Since our attack scheme has attack seed samples, the construction process of the problem space becomes simple. The main cost lies in how to confuse a sample after modifying a single-bit feature space. There are many mature code obfuscation tools available to meet this type of demand. We tested mainstream tools and found that the average processing time is 75 seconds per tool. The average size of the samples is 199.72MB, and the sample list is shown in Appendix E. In addition to the fact that automated tools in the industry have significantly reduced the cost of constructing poison samples for attackers, existing academic research has also shown that related construction is feasible, with the construction time for a single sample being 10 minutes per sample [16].

G Attack Value Unstable

Finally, it is important to note that the CDAD attack framework is an automated generation framework for concept drift poisoning samples. Therefore, the malware attack value assessment module and the survival time extension module need to be integrated with each model cycle. Furthermore, the attack value of high-value new malware in the initial model cycle is also unstable in subsequent model cycles. 85% of the samples with high attack value remain high-value in the subsequent model cycles, but nearly 15% of the samples experience fluctuations in attack value. Specifically, the attack value of the samples may oscillate between high and low values, or it may shift from high value to low value. To ensure the consistency of attack effectiveness across consecutive model cycles, we employ freezing attacks during the low attack value model cycles of new malware. Freezing Attack involves selecting Benign attack seeds with the highest uncertainty score. During the attack sample generation phase, data augmentation based on sample replication strategies is applied to produce a batch of attack samples.

H Concept Drift Adaptation Strategy Settings

It is not enough to just enumerate the model structure and concept drift strategy. What is more critical is how to

combine the target model structure and concept drift adaptation strategy. Our combination strategy follows the principle of optimal configuration. For each target model architecture, we select the corresponding concept drift strategy based on the optimal combination method in existing research methods. The last setting about the target model is the initial state of the target model. Our setting in this study is to train the initial data for 1 year, then conduct concept drift adaptation attack verification, and use the monthly time window to evaluate and verify the effectiveness of the attack. The final combination is shown in Table 2.

I Other Attack Influencing Factors

The sample feature search space refers to the entire feature space that the attacker can perturb when generating attack samples based on the attack seed samples. In CDAD attack, the entire perturbation feature space formed after a single bit flip is performed on each attack seed sample. The larger the search space, the greater the probability that the attacker will find samples that meet the attack requirements. However, the larger the search space, the greater the cost the attacker pays in sample generation. In this part experimental evaluation, the search space was set to the full feature space, 90% feature space, and 80% feature space respectively to verify the impact of the search space size on the attack effect. We selected the best concept drift adaptation method as the attack target and set the label budget to 200. The experimental evaluation results are shown in Table 13.

Table 13: Search Space Influence Factor

Search Space(%)	**F1**	**FNR (%)**	**ASR (%)**
100	0.90	14.12	**89.44**
90	0.89	14.54	**87.93**
80	0.87	19.16	**91.28**

According to the experimental result data, we can see that the average ASR under different search space settings can reach 89.55%. The setting group with a 20% reduction in search space can still achieve an ASR of 91.28%. This fully demonstrates that even when the attacker has limited attack costs, they can still carry out effective attacks.

J Potential Ethical Concerns

The main purpose of our research is to evaluate the security of concept drift adaptation methods, as related methods have currently received attention from researchers. Attackers are motivated to exploit the vulnerabilities of concept drift adaptation methods to prolong the survival time of new malware, thereby gaining more benefits. Even though the intent is strict about evaluating the robustness of concept drift adaptation methods, potential ethical concerns are associated with our research. For example, attackers can leverage our methods to carry out attacks or enhance malware. Therefore, following previous research precedents [16, 27, 45], we will restrict code sharing to verified academic researchers.