CDAD: Concept Drift Adaptation Denial Attack in Android Malware Detection

Anonymous submission

Abstract

Machine learning-based Android malware detectors shows encouraging result. Unfortunately, with the evolution of malware, deployed detection models will soon become outdated, a phenomenon known as concept drift. To address this issue, concept drift adaptation strategies based on active learning are proposed to improve detection models' performance on malware classification. The core of existing concept drift adaptation lies in selecting high value samples during the testing phase and optimizing the model based on these samples.

In this paper, we design a new attack in which adversaries construct poisoned samples based on high-value benign samples, controlling sample selection and model updating. The new attack, dubbed as concept drift adaptation denial (CDAD) attack, can prolong the survival time of new malware. We evaluate the CDAD attack on Android malware detection models, employing four concept drift adaptation strategies, and verify its attack effectiveness on a dataset comprising over 580,000 samples spanning more than 10 years. Experimental results indicate that CDAD achieves an attack success rate of 88% across four mainstream concept drift strategies under the white-box threat model and an attack success rate of 82.95% under the black-box threat model. Furthermore, the impact on the original model's performance during the attack process is minimal, with an average reduction of less than 0.02 F1, demonstrating high stealthiness in CDAD attack.

1 Introduction

The 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% [37], with nearly 4 billion active users worldwide [12]. Unfortunately, mobile devices and applications powered by the Android operating system have been selected as valuable targets by cyber criminals [5]. According to a security analysis by Kaspersky, even the official Google Play Store had over 600 million malware downloads in 2023 [41]. Facing the massive amount of malware generated daily, researchers have proposed automatic detection detectors for Android malware based on machine learning [48].

However, deploying Android malware detectors in the real world faces many challenges. One of the most critical challenges is that real-world data distribution can change over time, yielding the phenomenon of concept drift [6, 10, 15, 33, 44, 46]. Researchers have demonstrated that Android malware detectors, which performs well on training datasets, experiences a decline in its F1 score from 0.99 to 0.76 within approximately 6 months when faces with concept drift [10]. A solution is to add new data to the training dataset to ensure that training dataset distribution is consistent with the real-world data distribution [14]. But the number of new Android applications in the real world is overwhelmingly large. Google Play launched 1069 mobile apps every day in 2024 [7]. Therefore, obtaining sample labels for all new data is infeasible, which leads to an insufficient quantity of training datasets, resulting in a decline in the performance of Android malware detectors.

Existing solutions [6, 10, 15, 46] 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 [6, 46] has devised diverse evaluation methods aimed at assessing the value of test data. Then researchers introduce high-value samples to mitigate model performance degradation caused by concept drift. But we find that most previous research [6, 10, 15, 46] focus 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 on the vulnerabilities of Android malware detection either focuses on the security of training phase (poisoning attacks [13, 21, 27, 34, 45]) or the security of inference phase (evasion attacks [4, 20, 29, 32]). And we note that the concept drift adaptation focuses on the concept drift samples selection, which is different from the training and

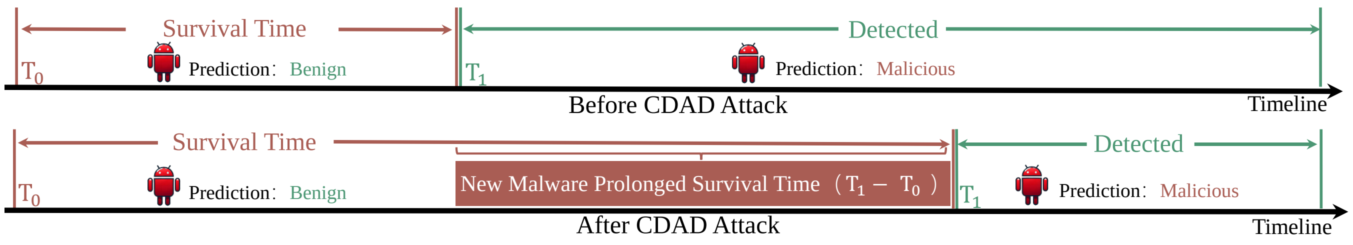

Figure 1: Prolonged survival time of new malware under CDAD attack

inference phase. 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 survival time of new Android malware under the concept drift adaptation methods. we apply concept drift adaptation methods [6, 10, 15, 46] against the new Android malware samples. We choose Android malware because of the availability of public, large-scale, and timestamped datasets(e.g., AndroZoo [1–3, 10, 49]). We find that some new malware samples have longer survival time than other malware samples. For example, the survival time of the tascudap family reaches over 2 years. Among the new malware families each month, over 95% of them contain samples with a survival time greater than 0 month. For detailed data on the survival time of new malware, refer to Figure 7 in Appendix C. This implies that there are vulnerabilities in the current concept drift adaptation strategies.

Based on the above insights, we propose the concept drift adaptation denial (CDAD) attack (§3) to prolong the survival time of new malware. CDAD attack can efficiently generate poisoned samples with clean labels. Our attack framework comprises three modules: surrogate model training, malware attack value assessment and malware survival time prolongation. We use the Attack Success Rate (ASR) to measure the attack effectiveness, which refers to the proportion of samples whose survival time has been effectively prolonged among all attacked samples. We conduct our evaluation under both white-box (§4.2.1) and black-box threat models (§4.2.3) and analyze the attack influencing factors (§4.2.2).

To better demonstrate the vulnerability of mainstream concept drift adaptation methods to CDAD attacks, we conduct experiments on four mainstream Android malware concept drift detection strategies, including Hierarchical Contrastive Classifier (HCC) [10], Transcending (TRANS) [6], high-dimensional outlier distance (CADE) [46], 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 attack success rate of 92.77% on the latest Strategy (HCC).

Our Contributions. To sum up, we mainly make the following contributions:

- Our research exposes the poisoning vulnerabilities in the concept drift adaptation strategies. The survival time of new malware can be prolonged by our CDAD attack.
- We propose an automatic poisoned sample generation framework for CDAD attack. This framework generates poisoned samples with clean labels, and it does not require any modifications to new malware samples. Therefore, our framework reduces the attack cost and enhances the stealthiness of our CDAD attack.
- We conduct CDAD attack effectiveness tests on two Android malware concept drift dataset over a period of 10 years, and conduct detailed discussions on 3 primary attack influencing factors, attack stealthiness and the costs of attackers and defenders.
- We provide an open source implementation of CDAD[1] attack to benefit future research in the community and encourage further improvement on our approach.

2 Background

2.1 Concept Drift Adaptation Paradigm

Different concept drift adaptation strategies share same paradigm (as shown in Figure 2). The classical paradigm consists of three stages. Stage one is test dataset inference. New Android applications are submitted to the detection model for testing. The detection model gives a predicted label. In the case of binary classification, the return label value is 0 or 1. Stage two is concept drift 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 design different sample selection strategies to find the most helpful samples for improving the model's performance under limited label budget. Stage three is incremental training dataset retraining. The model trainer adds selected samples

[1]CDAD is available at https://anonymous.4open.science/r/Denial-of-Concept-Drift-Adaptation-Code-C0EF

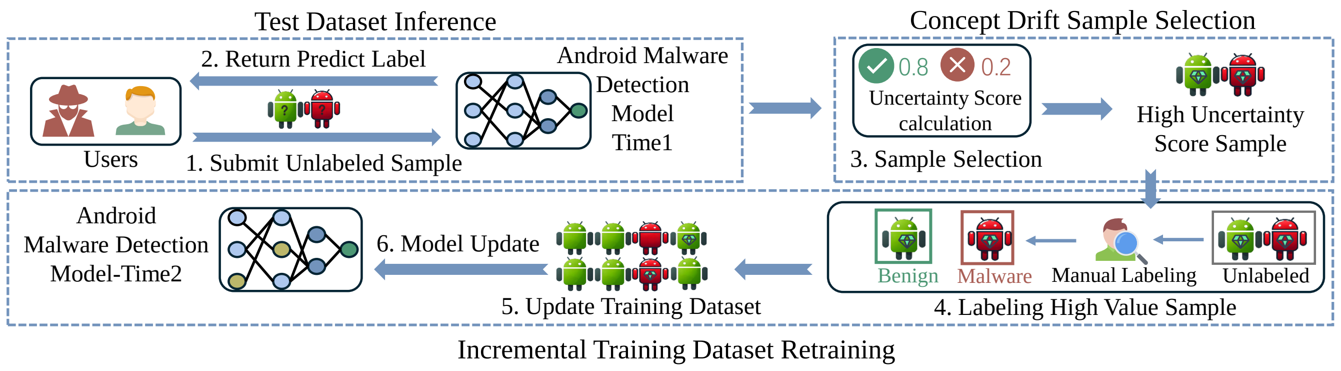
Test Dataset Inference
2. Return Predict Label
Android Malware Detection Model Time1
Users
1. Submit Unlabeled Sample
Concept Drift Sample Selection
0.8
0.2
Uncertainty Score calculation
High Uncertainty Score Sample
3. Sample Selection
Android Malware Detection Model-Time2
6. Model Update
5. Update Training Dataset
Benign
Malware
Manual Labeling
Unlabeled
4. Labeling High Value Sample
Incremental Training Dataset Retraining

Figure 2: Concept drift adaptation based active learning

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

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 gives priority to selecting the samples with high uncertainty of the current model for labeling. A common uncertainty measurement for a neural network is to use one minus the max softmax output of the network.

Encoder Space Distance. CADE [46] 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 training dataset.

Credibility and Confidence. Transcending [6] introduced the thery of conformal prediction [42] (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.

Hierarchical Contrastive Loss. The method proposed by Chen et al. [10] is currently the best-performing strategy in Android malware concept drift adaptation methods. The model consists of 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. The advantage of this strategy is its provision of finer-grained encoder rules and utilization of similar features among malware families, ultimately enhancing concept drift adaptation performance.

2.3 Attacks On Android Malware Detection

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 [9,22,31,50]. Specifically, the attacker's goal in an evasion attack is to add a small perturbation to a target malware 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 [21, 34]. These attacks assume attackers can inject poisoned samples into the training dataset. In poisoning attacks, the adversary's goal is to degrade model performance 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 victim model, poisoning attacks can be roughly divided into untargeted and targeted poisoning attacks. The goal of untargeted poisoning attacks is to decline the overall performance of the victim model. The goal of targeted poisoning attacks is to force the victim model to perform abnormally on a specific input class. Backdoor attacks [13, 21, 34] are a special case of targeted poisoning attacks where the victim model 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.

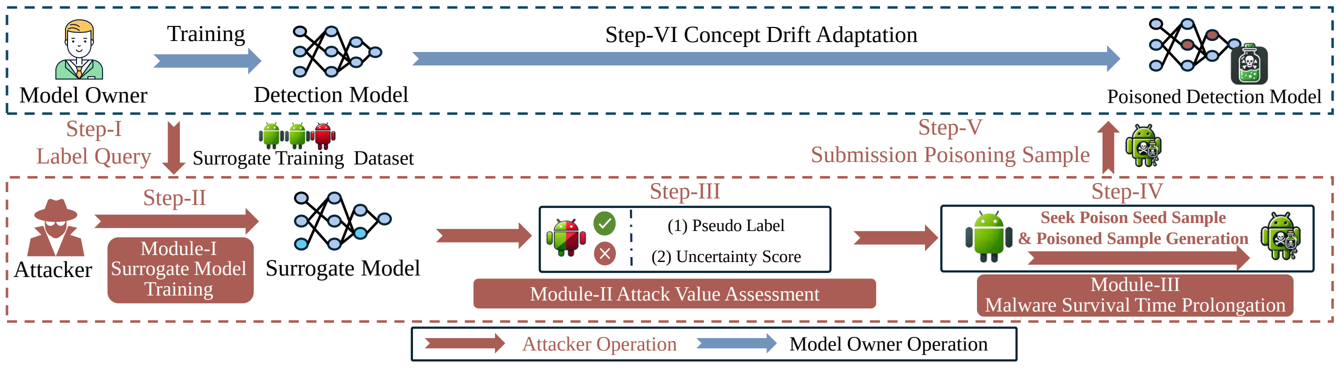
Model Owner
Training
Detection Model
Step-VI Concept Drift Adaptation
Poisoned Detection Model
Step-I
Label Query
Surrogate Training Dataset
Step-V
Submission Poisoning Sample
Attacker
Step-II
Module-I Surrogate Model Training
Surrogate Model
Step-III
(1) Pseudo Label
(2) Uncertainty Score
Module-II Attack Value Assessment
Step-IV
Seek Poison Seed Sample & Poisoned Sample Generation
Module-III Malware Survival Time Prolongation
Attacker Operation
Model Owner Operation

Figure 3: Concept drift adaptation denial (CDAD) attack

3 Attack Methodology

3.1 Threat Model

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 victim model's active learning incremental training dataset, concept drift adaptation strategies, sample feature vectors, and model parameters. This setting follows Kerckhos' principle [29], ensuring that the security of the model does not rely on secrecy. Additionally, concept drift adaptation strategies may ultimately favor some state-of-the-art methods [11], it will be challenging to maintain concept drift adaptation strategies strict confidentiality. In a black-box threat model [54], the attacker cannot obtain the training dataset or model parameters from the victim model. Therefore, the attacker can only rely on a surrogate model for approximate analysis, as demonstrated by previous work [39, 52].

3.2 Adversary's Challenges

Although attackers have some capabilities and knowledge (mentioned in §3.1), attacks against concept drift adaptation still face severe challenges. These challenges make our CDAD attack different from previous attacks (as shown in Table 1). The detailed challenges faced by attackers are as follows:

Table 1: Adversarial setting
(✓:have this characteristic,✗:lack of this characteristic)

Attack Method	**AC**	**TC**	**MI**	**LC**
Android HIV [9]	✗	✗	✗	✓
HRAT [50]	✗	✗	✗	✓
Severi et al. [34]	✗	✗	✗	✓
Li et al. [21]	✗	✗	✗	✓
AdvDroidZero [17]	✗	✗	✗	✓
Jigsaw Puzzle [45]	✗	✗	✗	✓
Our Work (CDAD)	✓	✓	✓	✓

Attack Continuity (AC). Previous poisoning attacks [18, 25, 40, 47] 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 challenging. This paper is the first to study the continuity of poisoning attacks on the concept drift adaptation process for Android detection models.

Training out of Control (TC). Even under the white-box threat model, the attacker cannot directly poison the training dataset. They only have sample submission and query access to the latest state of the victim model. Compared with the attack challenge of existing attack scheme [34, 38], our attacker challenge has a higher degree of difficulty.

Malware Integrity (MI). Although some attackers have used code updates to counter the detection methods [8, 9, 23, 36]. However, this method requires attackers to pay expensive costs for code development. Furthermore, frequent code updates by attackers will lead to changes in malware hash, which will further cause the detection engine to constantly detect modified malware.

Label Correctness (LC). The labels of poisoned samples uploaded by the attacker will not be mislabeled, which is different from the assumption of many advanced model attacks in the image field [16, 51]. The reason is that sample labeling in the active learning process is done manually. This is also a special challenge in our CDAD attack.

3.3 Attack Method

Taking the above attack challenges as the prerequisite, we propose concept drift adaptation denial (CDAD) attack. The overall workflow is shown in Figure 3. The first module is to conduct a surrogate model, which is responsible for simulating the target model and providing a basis for subsequent attack steps (§3.3.1). The target model refers to the victim model mentioned previously. The second module is malware attack value assessment, which is responsible for assessing the attack value of new malware samples and selecting attack strategy (§3.3.2). The third module involves the generation of poisoned samples (§3.3.3). To clearly describe our CDAD attack, we have introduced relevant symbols. Please refer to Appendix A for a complete list of symbols and their meanings.

3.3.1 Surrogate Model Training

We build a surrogate model for obtaining 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.

Considering that our attack scheme has obvious temporal characteristics, we use i to uniformly represent different model update time nodes. In the initial state, both the surrogate model and the target model are trained based on their respective initial training datasets. Due to the openness of Android data collection, the initial training datasets for the surrogate model and the target model are consistent. Therefore, we also set the initial parameters θ_a^0 of the surrogate model and the initial parameters θ_d^0 of the target model to be consistent. Differences between the surrogate model and the target model come from the model retraining stage. At time node i, regarding training data acquisition, the owner of the target model is always a large security vendor, so it can effectively collect new Android samples D_t^i and detect concept drift samples D_c^i. More importantly, the owner of the target model has the ability to conduct reliable sample label analysis on concept drift samples D_c^i.

However, as attacker, we lack the ability to provide reliable labels for concept drift samples. We collect new data samples D_t^i from the real world, identifies concept drift samples $\bar{D_c^i}$, and obtains query results as pseudo labels for concept drift samples $\bar{D_c^i}$ based on the target model θ_d^i. It is important to emphasize that due to the openness of the Android platform, our ability to collect data aligns with the owner of target model, so we all get Android samples D_t^i. Additionally, our purpose is to approximate the detection ability of the target model θ_d^i, so we do not need to care about the true label of the concept drift samples $\bar{D_c^i}$ but only needs to get the sample prediction result (pseudo label) of the target model θ_d^i, which greatly reduces our label cost. Then, the surrogate model θ_a^i is retrained based on concept drift samples $\bar{D_c^i}$ 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, our surrogate model must also be constantly updated. We define the time interval between the two model update time nodes as a unit model period. In a unit model period (at time node i), 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, we query 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 with these samples. By repeating the above process, we 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 approximation of the detection ability of the surrogate model to the target model.

Algorithm 1: Surrogate model training

```
1  Input: time node $i$, new Android samples $D_t^i$, target
          model $\theta_d^i$, concept drift adaptation label budget
          $LB$, loss function $L$ for model training process,
          training dataset $D_T$.
2  Output: Next time node ( $i+1$ ) surrogate model
           sequence $\theta_a^{i+1}$.
3  Step I: Concept Drift Samples Detection
4  calculate $D_t^i$ uncertainty scores $U_t^i$  ▷ As defined in 2.2
5  $U_t^i \leftarrow f_c(D_t^i)$
6  Step II: Concept Drift Samples Selection
7  get concept drift samples $\bar{D_c^i}$ (within the label budget
     $LB$ ) based on the uncertainty score $U_t^i$
8  $\bar{D_c^i} \leftarrow SamplesSelection(D_t^i, LB)$
9  Step III: Pseudo Label Query
10 query the target model $\theta_d^i$ for the pseudo labels $\hat{Y}_i$ of
     concept drift samples $\bar{D_c^i}$
11 $\hat{Y}_i \leftarrow Query(\bar{D_c^i}, LB)$
12 Step IV: Surrogate Model Retraining
13 add concept drift sample data $\bar{D_c^i}$ to the surrogate
     model training dataset $D_T$
14 $D_T \leftarrow UpdateDataset(D_T, \bar{D_c^i})$
15 retrain the surrogate model $\theta_a^i$ based on the surrogate
     model training dataset $D_T$ to obtain a new surrogate
     model $\theta_a^{i+1}$
16 $\theta_a^{i+1} \leftarrow Training(D_T, L, \theta_a^i)$
17 return $\theta_a^{i+1}$
```

Algorithm 1 shows the 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.

3.3.2 Malware Attack Value Assessment

Based on the surrogate model, we have the ability to evaluate the attack value of new malware samples, and then provide differentiated attack strategies. The malware attack value assessment is specifically divided into two sub-modules, which assess the attack value of new malware samples in the current model cycle and the future model cycle, respectively.

1) attack value assessment based on model prediction results: This part is used to measure the likelihood of new malware surviving in the current model cycle. Specifically, we submit the new malware sample x_m to the surrogate model θ_a^i at time node 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 samples x_m at current model cycle is low.

2) attack value assessment based on uncertainty score: Although attack value assessment based on model prediction results can eliminate samples with no attack value, it cannot prioritize the samples with attack value. Therefore, we propose the uncertainty score attack value assessment. It takes the survival probability in the next model cycle as the basis for judging the attack priority of malware x_m. Based on the characteristics of the concept drift adaptation process, the survival probability of new malware in the next model cycle is usually affected by sample selection caused by the concept drift adaptation strategies. Therefore, when the uncertainty score of new malware x_m is lower than the lowest score u_{min}^i for sample selection in the current model cycle (time node i), 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 (§4.3).

3.3.3 Malware Survival Time Prolongation

Next, we introduce the malware survival time prolongation module, which is the core of the CDAD attack framework. Malware survival time prolongation mainly includes three parts, namely poisoned seed sample selection, poisoned sample generation, and poisoned sample data distribution disturbance.

3.3.3.1 Poisoned Seed Sample Selection We abstract the 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 (D_t^i) collected by us. 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 CDAD attack are all benign samples, which provide the basis for clean label attacks.

$$X = \{x | x \in (D_T \cup D_t^i) \wedge (y_x == 0)\} \tag{1}$$

We define all the data available to us (including the existing surrogate training dataset D_T and the new Android dataset D_t^i at time node i) as the search space for identifying poisoned seed samples. Let x represent a sample in the search space and y_x represent the label of sample x. X represent the set of benign samples in the search space (Equation 1).

$$X_s^i = \{x_k | f_c(x_k) > u_{min}^i \wedge x_k \in X\} \tag{2}$$

We select sample x_k with uncertainty scores $f_c(x_k)$ greater than the minimum sample selection uncertainty score u_{min}^i at the current time node i as the seed samples X_s^i for CDAD attack (Equation 2).

3.3.3.2 Poisoned Sample Generation Poisoned seed samples have the capability to influence the decision boundary of the target model. However, since the search for seed samples is based on optimization strategies, meeting the required capacity for poisoned attack samples is challenging. The range of sample selection in concept drift adaptation typically involves a set of samples rather than a single sample (Chen et al. [10] set label budget as 50, 100, 200 and 300 samples). Moreover, the stronger the capability of the target model, the better its sample labeling capacity, resulting in a larger range for sample selection. It is difficult for too few poisoned samples to have a big impact on the target model. Therefore, the challenge faced by us is how to generate a large number of effective poisoned samples based on a small number of poisoned seed samples. It is important to note that the generation of poisoned samples is called the inverse mapping problem of feature space in problem space in existing research [29]. That is to say, we need to write an actual program in the corresponding problem space [2] based on the given features. Therefore, poisoned sample generation consists of two parts. The first part is feature vector construction of poisoned samples, which aims to find samples that meet the attack requirements. The second part is feature space inverse mapping, illustrates that the feature space of the poisoned 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. [29], so our attack approach primarily focuses on constructing the feature space for poisoned samples.

Feature Vector Construction: To maintain the effectiveness of the attack, we adhere to the following three constraints during the feature vector construction process.

constraint 1: The poisoned samples constructed from the seed samples must have labels (benign) 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 poisoned 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

[2]The problem space refers to the set formed by all Android Application Packages (APK)

range sample, thereby ensuring that the poisoned 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 poisoned sample is higher than that of the sample (x_m) to be protected and does not consider other samples. Therefore, it is a targeted poisoning attack among poisoning attacks.

Based on the aforementioned three constraints, we propose a label consistency data perturbation method. The specific steps are shown in Algorithm 2.

Algorithm 2: Malware Survival Time Prolongation

```
1  Input: time node $i$, poisoned seed samples $X_s^i$,
          surrogate model $\theta_a^i$, 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 $x_m$
2  Output: poisoned samples $D_p^i$.
3  Step I: Feature Space Perturbation
4  while keeping the label stable, perform feature
     perturbation on samples $X_s^i$, form perturbed datasets
     $D_s^i$.
5  for each $k \in len(X_s^i)$ do
6      $D_{sk}^i \leftarrow f_p(x_s^k)$                    ▷ constraint 1
7  $D_s^i \leftarrow [D_{s0}^i \cup D_{s1}^i \cup .....D_{sp}^i]$ $(p = len(X_s^i) - 1)$
8  Step II: Uncertainty Score Evaluation
9  perturbed datasets $D_s^i$ are submitted to the surrogate
     model $\theta_a^i$ for testing, and the surrogate model feeds
     back the uncertainty score
10 for each $j \in len(D_s^i)$ do
11     $u_j \leftarrow f_c(p_s^j, \theta_a^i)$, $p_s^j \in D_s^i$
12     if $u_j > u_{min}^i$ then
13         $D_s^i \leftarrow (D_s^i - p_s^j)$              ▷ constraint 2
14     if $u_j < u_m$ then
15         $D_s^i \leftarrow (D_s^i - p_s^j)$              ▷ constraint 3
16 $D_p^i \leftarrow D_s^i$
17 return $D_p^i$
```

To satisfy the label constraint mentioned in constraint 1, we adopt a specific perturbation method for the feature perturbation function (f_p) in Algorithm 2-Step I. This method involves positive flipping for malware sample feature space and negative flipping for benign sample feature space.

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^i can be used as poisoned samples for CDAD attack.

At this time, the generated batch poisoned samples D_p^i 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 poisoned samples effectively change the model decision boundary into a state favorable to the malware sample x_m.

3.3.3.3 Poisoned 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 victim's suspicion. To further enhance the stealthiness of the attack, we need to increase the distribution diversity of the sample selection data used in our CDAD attack. 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 analyze 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.

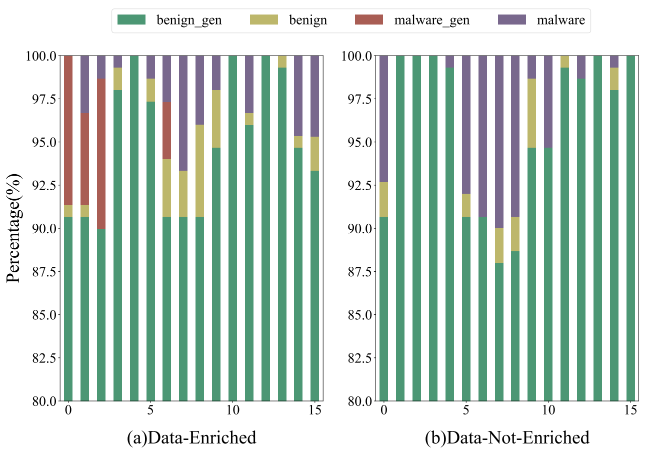

Figure 4: Poisoned sample data distribution disturbance

These malicious samples are all from the training data used for the target 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 our CDAD 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 poisoned sample generation algorithm. This simply requires replacing the input seed data, as demonstrated in Algorithm 2.

As shown in Figure 4, we conducted statistics on the data distribution of attack samples (HCC as attack target and label budget is 150 samples). Figure 4-a shows the data distribution of concept drift samples after introducing the perturbation module for attack sample data distribution. In contrast, Figure 4-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 stealthiness of attack behaviors.

4 Empirical Evaluation

4.1 Experimental Settings

Datasets. Android malware concept drift dataset is a special type of dataset with a time attribute. Compared with classic Android malware datasets, such as Drebin [5], CIC2020 [19, 30], etc., concept drift datasets help researchers to do more in-depth research on malware evolution characteristics. The construction of a concept drift dataset requires extensive and long-term collection of Android software. So there are few concept drift datasets. Researchers often need to collect the data themselves or perform secondary processing on existing non-concept drift Android malware datasets.

In this study, we select two Android malware concept drift datasets (APIGraph [49] and CDAMAL[3]) to validate the effectiveness of our attack methods. Both datasets are characterized by long-term collection and rich family information. We collect and construct the CDAMAL concept drift adaptation dataset. Details of the collection and construction process are in Appendix B. The combined time span of both datasets covers more than 10 years. The number of samples in the dataset reaches 580,000, with more than 1000 malware families.

Target Model Settings. Based on the aforementioned large-scale concept drift datasets, we set our attack objectives. Specifically, attack objectives include the configuration of the model structure and the implementation of adaptation strategies for concept drift. For the target model setup, we base on the current best practices in the field of concept drift adaptation (HCC [10]) and enrich the target model with the model mentioned in He et al. [17] and Fang et al. [14]. The specific model setting combination is shown in Table 2. For more details about concept drift adaptation strategy settings, refer to Appendix D. In setting up the concept drift adaptation strategies, we test not only the current optimal strategy (HCC [10]) but also several key milestone approaches (TRANS [6], CADE [46],UNCER [15]).

Table 2: Combination of model structure and CDA-strategy

Strategy	Structure
HCC [10]	Encoder-Classifier (MLP) [10]
TRANS [6]	Classifier (SVM) [5]
CADE [46]	Encoder-Classifier (MLP) [28]
UNCER [15]	Classifier (ResNet) [14]

Surrogate Model Settings and Evaluation Matrix. Based on the target model settings, 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 §3.1. 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 §4.2.3. In terms of performance indicators, we use F1 score, false negative rate (FNR) and other indicators to measure model performance. Regarding attack effectiveness metrics, we report 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 our CDAD attack.

Attack Objective Settings. To verify the effectiveness of CDAD attack, we conduct active learning tests on the existing concept drift adaptation strategies. The test results and model hyperparameter settings provided in Appendix E. Then we select new malware samples from new family or new varients of old family as the attack 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%. Secondly, these samples have consumed the attacker's development costs but have not provided benefits to the attacker. So attackers have the motivation to prioritize attacking these samples. Based on the aforementioned strategy for selecting attack targets, we extract samples from new malware families for validation of the attack method.

4.2 Attack Effectiveness

Based on the above settings, we evaluate the effectiveness of our CDAD attack. The evaluation of attack effectiveness is conducted under both white-box and black-box threat models. In order to study the impact of different settings on the

[3]CDAMAL dataset is available at https://anonymous.4open.science/r/CDAMAL-Concept-Drift-Dataset-3E08/

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

Concept Drift Strategy	F1-Score	TPR (%)	FPR (%)	TNR (%)	FNR (%)	ACC (%)	PREC (%)	ASR (%)
HCC	$0.90_{-0.02}$	$85.88_{-2.80}$	$0.44_{+0.04}$	$99.56_{-0.04}$	$14.12_{+2.80}$	$98.27_{-0.28}$	$95.42_{-0.46}$	**92.77**
TRANS	$0.89_{-0.01}$	$85.18_{-1.12}$	$0.53_{-0.01}$	$99.47_{+0.01}$	$14.82_{+1.12}$	$98.09_{-0.11}$	$94.52_{+0.12}$	**89.92**
CADE	$0.88_{-0.02}$	$85.57_{-2.62}$	$0.90_{+0.06}$	$99.10_{-0.06}$	$14.43_{+2.62}$	$97.79_{-0.33}$	$91.39_{-0.68}$	**86.90**
UNCER	$0.88_{+0.01}$	$83.86_{+0.98}$	$0.65_{+0.00}$	$99.35_{+0.00}$	$16.14_{-0.98}$	$97.88_{+0.11}$	$93.22_{+0.08}$	**82.86**

effectiveness of our attack, we analyze various influencing factors.

4.2.1 White Box Attack Effectiveness

Strong attackers have white-box access to model information. We evaluate CDAD attack on 4 models and the corresponding concept drift adaptation strategies. We set the label budget as 200 samples, which represents the average labeling capability of target model. The attack experimental result data is shown in Table 3.

First, we observe that the poisoned samples significantly degrade the target model's ability to detect new malware. Our proposed attack method can achieve effective attacks, with an average ASR reaching 88%. The ASR can reach 92.77% when targeting the currently optimal method (HCC) for concept drift adaptation. The main reason why CDAD attacks achieve the highest attack success rate on HCC is that HCC selects concept drift samples based on sample loss function values, so the constructed attack samples have a greater impact on the retraining process of the target model.

At the same time, we note the attack's impact on the detection performance of other non-targeted Android samples. The purpose of targeted poisoning attacks carried out by attackers is to prolong the survival time of new targeted malware. Therefore, while ensuring the success of the CDAD 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 select F1 score as the performance metric for target 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 stealthiness of the CDAD attack. The F1 performance indicator varies within 0.02. Even under the UNCER setting, the F1 performance has a slight increase compared with concept drift adaptation process without attack, indicating the stealthiness of our attack. From the user's perspective, the average TNR of the model under CDAD 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 poisoned samples can also benefit from the attack. We analyze 57 successful attack samples from new families under the pseudo-loss setting and find 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. The survival time of these malware samples will be prolonged.

4.2.2 Attack Influencing Factors

So far, we have shown that our proposed CDAD attack is effective against a large amount of new malware samples. Then we conduct a further analysis to investigate the impact of variations in CDAD attack scheme settings on the overall 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: the label budget, the proportion of the label budget occupied by attackers, and the various feature extraction methods employed.

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 5. It can be seen that no matter how much the label budget is, the attack is effective. And we note that as the label budget increases, the attack effect also improves synchronously. The setting of a label budget of 300 samples increased by 5.74% compared to the setting of a label budget of 50 samples. 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 poisoned samples within the label budget represents the intensity of the CDAD attack. The higher the proportion

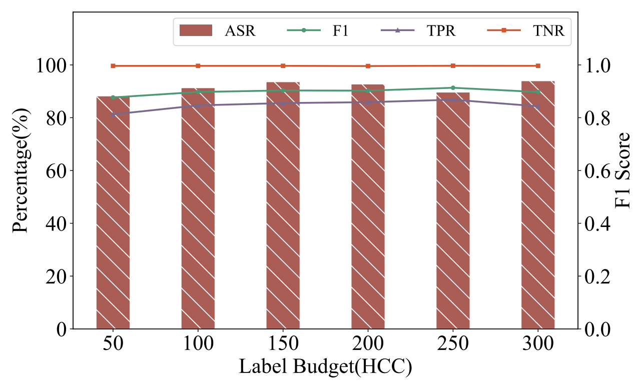

Figure 5: Attack effectiveness under different label budget

of poisoned 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% and 50% respectively. We then evaluate how different label budget proportions affect the attack result. As shown in Table 4, different label budget proportions have different impacts on ASR. The average ASR of multiple sets of attack tests can still reach 87.73%. Furthermore, 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 3.88% and 11.25% in ASR, respectively.

Table 4: Proportion of label budget occupied by attacker

Proportion	F1	FPR (%)	FNR (%)	ASR (%)
100	0.90	0.44	14.12	**92.77**
70	0.90	0.43	14.22	**88.89**
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 extract static features [5], 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 [49]	$0.90_{-0.02}$	$14.12_{+1.12}$	**92.77**
Drebin [5]	$0.67_{+0.01}$	$44.57_{+1.39}$	**95.03**

4) Search Space: 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 is 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, a larger search space will result in higher sample generation costs for the attacker. In this experimental evaluation, the search space was set to 100% feature space, 90% feature space, and 80% feature space respectively to verify the impact of the search space size on the attack effect. We select the best concept drift adaptation method (HCC) as the attack target and set the label budget to 200 samples. The experimental evaluation results are shown in Table 6.

Table 6: Search space influence factor

Search Space (%)	F1	FNR (%)	ASR (%)
100	0.90	14.12	**92.77**
90	0.89	14.54	**92.10**
80	0.87	19.16	**95.93**

According to the experimental result, we can see that the average ASR under different search space settings can reach 93.60%. The setting group with a 20% reduction in search space can still achieve an ASR of 95.93%. These results demonstrates that when the attacker cannot search the entire feature search space due to the limited attack cost, they can still carry out our CDAD attack.

4.2.3 Black Box Attack Effectiveness

To demonstrate the effectiveness of CDAD attack under the black-box threat model, we have set up the role of a weak attacker. Specifically, compared to the strong attacker setting, we adjust some settings 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[4] the model settings for the attacker. Because of the current optimal concept drift adaptation methods mainly consist of an encoder (ENC) and a classifier (CLA), we provide four sets of comparative settings based on the target model (as shown in Table 7).

Table 7: Attack effectiveness under model heterogeneity

Model	F1	ASR (%)	R-ASR (%)
ENC-CLA	0.89	**82.95**	**82.95**
ENC (Weak)-CLA	0.87	**72.31**	**59.09**
ENC-CLA (Weak)	0.83	**82.22**	**44.31**
ENC (Weak)-CLA (Weak)	0.86	**87.18**	**39.77**

Previous research has indicated that clean-label attacks suffer from end-to-end performance degradation [43], as model updates can lead to a deterioration in attack effectiveness.

[4]The specific setting for model weakening is to reduce the number of neural network layers in the encoder or the classifier.

However, the CDAD attack alleviates the issue of diminished attack effectiveness under end-to-end conditions. The experimental result data in Table 7 shows that the average ASR under various black-box settings reaches 81.17%. 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 attack 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 method.

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

$$R-ASR = \frac{NSAS(Weak-Settings)}{NSAS(ENC-CLA)} \tag{3}$$

Additionally, 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 analyze the selection of attack targets under different settings. We find that while the synchronized weakening setting exhibits an advantage in terms of the ASR metric, it demonstrates a 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 take into account both 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 (Equation 3), 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.

4.3 Attack Value Assessment Necessity

Analysis of the attack value of new malware samples provides support for subsequent survival time extension modules. In order to illustrate the role of this part, we conduct relevant ablation experimental analysis. Specifically, attackers skip the attack value assessment phase and directly move to the seed sample selection and poisoned sample generation phases. This means that attackers operate on all new malware, significantly increasing the cost of attacks.

Table 8: Attack value assessment necessity analysis

Stratege	Ablation-ASR (%)	ASR (%)
HCC (Model Based)	81.73	**92.77**$_{+11.04}$
CADE (Data Based)	72.43	**86.90**$_{+14.47}$

Our ablation study (shown in 8) involving 2 different concept drift strategies and label budgets (200 samples), the absence of the attack value assessment module led to an average decrease of 12.76% in attack success rates, which demonstrates that the attack value assessment module is an essential part of CDAD attacks. It should be noted that HCC and CADE are selected as experimental subjects in our ablation experiment because they represent two types of concept drift adaptation strategies respectively, which can fully demonstrate the value of the malware attack value assessment module.

4.4 Attack Explainability

This section further analyzes and explains why our CDAD attack is effective from the perspective of the model decision boundary. Taking the 'execdownload' family that emerges in the real world in June 2015 as an example, we illustrate the effectiveness of CDAD attacks (HCC as attack target and label budget is 200 samples). As shown in Figure 6-No Attack, 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 are 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 (as shown in Figure 6-CDAD Attack). The reason for the above phenomenon is that the change of the decision boundary depends on high uncertainty score samples. Although poisoned samples have high uncertainty score and include both benign and malware samples, their influence 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.

4.5 Attacker and Defender Cost Analysis

The defender's cost mainly comes from labeling new concept drift samples, which are selected during the sample selection stage of active learning. This cost depends on the size of

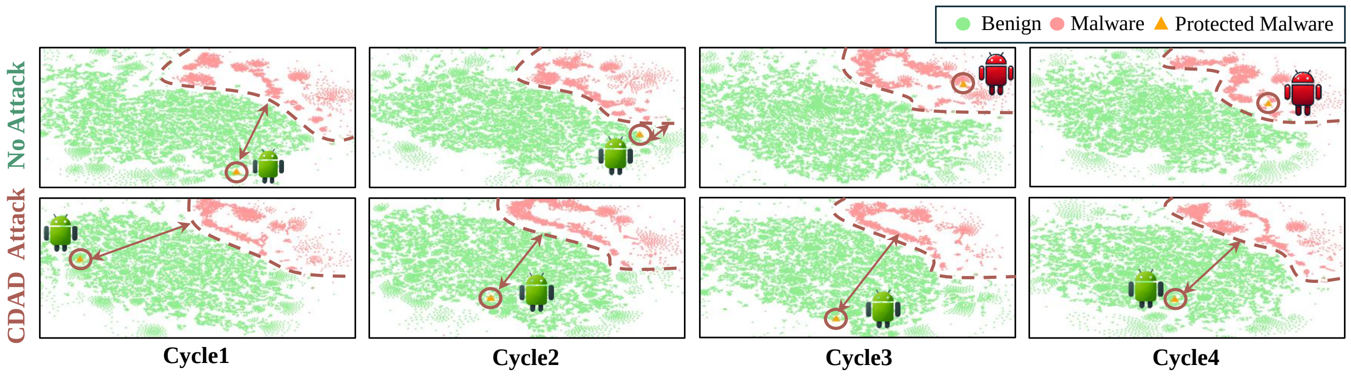

Figure 6: Attack explainability

the label budget. The attacker's cost structure is complicated, including manual labeling costs, attack seed sample search costs, and sample generation 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, 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 the construction of the poisoned sample. This part involves the construction of the corresponding problem space software programa 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 demand. We test mainstream tools and found that the average processing time is 75 seconds per sample. The average size of the samples is 199.72MB, and the sample list is shown in Table 9. In addition to the fact that automated tools in the industry have significantly reduced the cost of constructing poisoned samples for attackers, existing academic research has shown that related construction is feasible, with the construction time for a single sample being 10 minutes [17].

4.6 Time Cost of CDAD Attack

To fully demonstrate the rationality of attack in the problem space, we test the time cost of obfuscation operations.

Table 9: 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

Our purpose is to demonstrate that attackers can quickly map attack samples from the feature space to the problem space. We select APKs of different types and sizes. Then we test their repackaging and obfuscation time, as shown in Table 9. Based on the time-based test result, we can observe that the average attack time overhead for a single sample in the problem space is less than 5 minutes. Given that the concept drift adaptation model is typically updated monthly, attackers have sufficient time to execute CDAD attack. After fully confirming the effectiveness of the attack method, we conduct tests on the attack time cost, as shown in Table 10. We evaluat 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. Because of the different sizes of malware packages, the time cost for problem space attacks varies significantly. Therefore, we select 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.

5 Discussion and Future Work

In this study, we introduce an efficient framework for concept drift adaptation denial (CDAD) attack. 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 poisoned attack seed samples with high uncertainty scores. However, in some cases, such high uncertainty score samples may not appear in the test dataset of the current month or the attack value of new malware samples is unstable. To enable CDAD to adapt to such scenarios, we adopt strategies to relax the uncertainty score constraints or employ a freeze attack strategy. Refer to

Table 10: CDAD 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
Seed Selection (Min)	7.55	15.42	17.78	10.15	3.98	10.77	5.63	9.45	8.13	6.03	3.03	11.07
Sample Generation (Min)	1.7	3.58	4.37	2.08	1.47	5.52	1.23	2.52	2.27	1.85	1.07	4.35
Model Update (Min)	4.65	4.62	5.17	5.3	1.65	3.27	1.48	1.97	1.9	1.72	0.55	2.27
Stage	**TRANS**						**UNC**					
	50	100	150	200	250	300	50	100	150	200	250	300
Seed Selection (Min)	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 (Min)	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 (Min)	1.78	1.7	1.77	1.84	1.9	1.7	3.7	2.88	1.15	5.68	2.32	3.78

Appendix F for details about attack value unstable and freeze attack. Nevertheless, we acknowledge that freeze attack may potentially reduce the success rate or stealthiness of the attack. In future research, we plan to overcome these limitation.

Defense Strategy: As we know, current concept drift adaptation methods lack corresponding research on defense strategies. Only Lin et al. [26] conduct research on defense strategies against poisoning attacks in active learning. However, they assume that the attacker has the ability to maliciously mislabel sample label, while optimal concept drift adaptation 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.

6 Related Work

This work is broadly related to works on the survival time of Android malware and attack for Android malware detectors.

Malware Survival Time. Research on the survival time of malware primarily focuses on the dynamic changes of malware detection results and their influencing factors. Shuofei Zhu et al. [53] are 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 [53], our research focuses on the influencing factors of the survival time of new malware.

Attack for Android Malware Detectors. The attack methods targeting Android malware detectors are primarily categorized into evasion attack and poisoning attack. 1) Evasion Attack: Attackers can obtain false-negative detection results for new malware by performing operations such as repackaging and manipulating control flow graphs [9, 24, 35]. The currently mainstream method, proposed by Ping He et al. [17], can achieve effective attacks under a zero-knowledge setting. 2) Poisoning Attack: Apart from modifying malware, attackers have also proposed poisoning attack methods that exploit security vulnerabilities in the training phase of malware detection models [13, 21, 34]. Poisoning attacks can be divided into non-targeted poisoning attacks and targeted poisoning attacks (backdoor attacks). Currently, backdoor attacks are the primary form of poisoning attacks against Android malware detectors. The current mainstream method is to carry out backdoor attacks by adding triggers to new malware [45], thereby prolonging its survival time. However, both of the aforementioned methods require modifications to the new targeted malware, which increases the attack cost. Therefore, we propose CDAD attack, 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.

7 Conclusion

In this paper, we propose CDAD, a concept drift adaptation denial attack that achieves a high success rate in prolonging the survival time of target new malware, while exhibiting extremely high attack stealthiness. We design an automatic poisoned sample generation framework which generates poisoned samples with clean labels, and it does not require any modifications to new malware samples. Our large-scale experiments of a decade-long real-world dataset shows that current concept drift adaptation strategies are vulnerable to CDAD attack. Our work provides 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*, pages 468–471, 2016.

[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, 2018.

[5] 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.

[6] 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, 2022.

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

[8] Hamid Bostani and Veelasha Moonsamy. Evadedroid: A practical evasion attack on machine learning for black-box android malware detection. *Computers & Security*, 139:103676, 2024.

[9] 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.

[10] 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.

[11] Tianshuo Cong, Xinlei He, Yun Shen, and Yang Zhang. Test-time poisoning attacks against test-time adaptation models, 2023.

[12] coolest gadgets. Android statistics by users and revenue. https://www.coolest-gadgets.com/android-statistics, 2024.

[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*, 18:3977–3990, 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] Jonas Geiping, Liam Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching, 2021.

[17] 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.

[18] Wenbo Jiang, Hongwei Li, Guowen Xu, and Tianwei Zhang. Color backdoor: A robust poisoning attack in color space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8133–8142, June 2023.

[19] 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.

[20] 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.

[21] 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, 2022.

[22] 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.

[23] Heng Li, Zhang Cheng, Bang Wu, Liheng Yuan, Cuiying Gao, Wei Yuan, and Xiapu Luo. Black-box adversarial example attack towards {FCG} based android malware detection under incomplete feature information. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1181–1198, 2023.

[24] 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, 2020.

[25] 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, Anaheim, CA, August 2023. USENIX Association.

[26] 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, 2021.

[27] Faria Nawshin, Romain Arnal, Devrim Unal, Ponnuthurai Suganthan, and Lionel Touseau. Assessing the effect of model poisoning attacks on federated learning in android malware detection. In *Proceedings of the Cognitive Models and Artificial Intelligence Conference*, AICCONF '24, page 147–154, New York, NY, USA, 2024. Association for Computing Machinery.

[28] 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 Trans. Priv. Secur.*, 22(2), apr 2019.

[29] 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, 2020.

[30] 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.

[31] 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.

[32] 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.

[33] Jeffrey C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, September 1986.

[34] 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.

[35] Zhan Shu and Guanhua Yan. Eagle: Evasion attacks guided by local explanations against android malware classification. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3165–3182, 2024.

[36] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. Mab-malware: A reinforcement learning framework for blackbox generation of adversarial malware. In *Proceedings of the 2022 ACM on Asia conference on computer and communications security*, pages 990–1003, 2022.

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

[38] Bing Sun, Jun Sun, Wayne Koh, and Jie Shi. Neural network semantic backdoor detection and mitigation: A causality-based approach. In *Proceedings of the 33rd USENIX Security Symposium. USENIX Association, San Francisco, CA, USA*, 2024.

[39] 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.

[40] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Comput. Surv.*, 55(8), dec 2022.

[41] 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.

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

[43] 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 Comput. Surv.*, 55(7), dec 2022.

[44] Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23(1):69–101, April 1996.

[45] 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.

[46] 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.

[47] Ziqing Yang, Xinlei He, Zheng Li, Michael Backes, Mathias Humbert, Pascal Berrang, and Yang Zhang. Data poisoning attacks against multimodal encoders. In *International Conference on Machine Learning*, pages 39299–39313. PMLR, 2023.

[48] Yanfang Ye, Tao Li, Donald Adjeroh, and S. Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Comput. Surv.*, 50(3), jun 2017.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5251–5268, 2024.

A Notation

Symbol	Description
θ_d^i	Target model parameters at time node i
θ_a^i	Surrogate model parameters at time node i
D_c^i	Concept drift samples at time node i
D_t^i	Testing samples at time node i
$\hat{Y}_i$	Pseudo labels of concept drift samples D_c^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 at time node i
x_m	Protected malware sample (attack target)
$\hat{y}_m$	Predicted label of protected malware sample
v_m	Attack value of malware
u_m	Uncertainty score of protected malware sample
X_s^i	Selected poisoned seed sample at time node i
D_p^i	Generated poisoned samples at time node i
D_s^i	Perturbed dataset for Samples from X_s^i
M	Original feature space dimension of Samples

B Collection of Concept Drift Datasets

As shown in Table 11, our experimental evaluation dataset consists of two parts. The first part of the dataset is derived from APIGraph [49], while the second part is self-constructed. We will mainly elaborate on the self-constructed dataset.

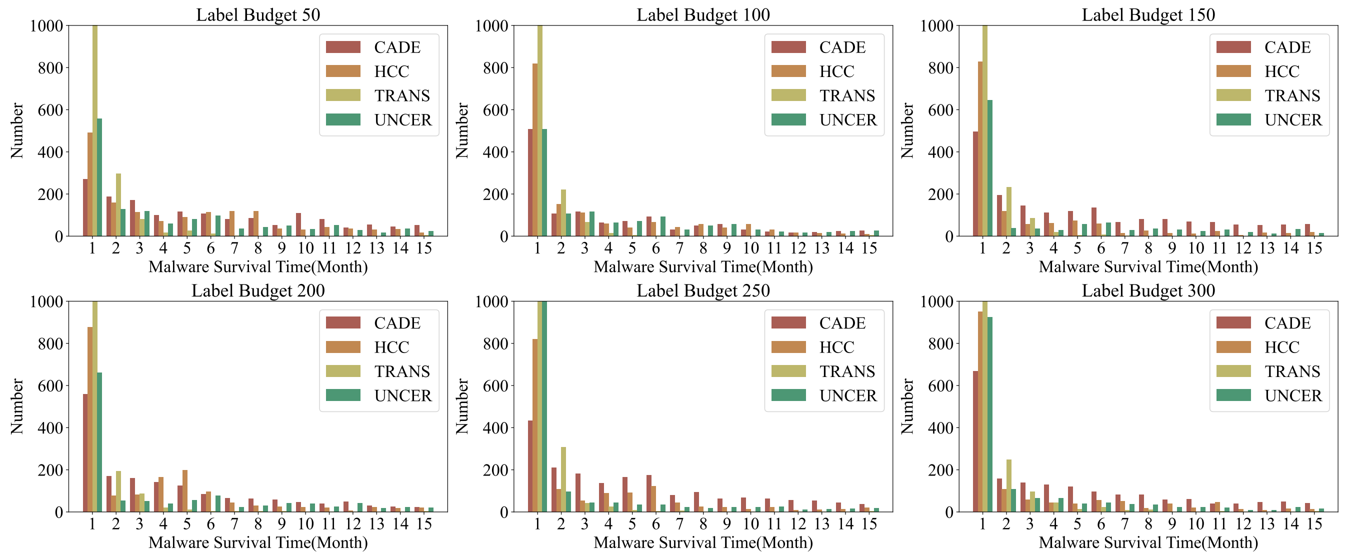

Figure 7: Survival time of new malware under concept drift adaptation strategies (with multiple label budget settings)

Table 11: 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 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. We classify the APKs into different time windows based on the "vt_scan_date" field in the CSV file, and categorize them as benign or malware based on the "vt_detection" field. For malware, we collect 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. Then we extract the APK features based on the static analysis method proposed by Rahali et al. [30], thereby generating the final Android malware concept drift dataset. The dataset spans six years and contains a total of 239166 benign samples, 26574 malware samples, and 1584 malware families. For detailed feature information, please refer to Table 12.

Table 12: Feature information

Feature	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		

C Original Survival Time of All Label 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 7. We can observe that the survival time of new malware exhibits a consistent trend, regardless of the label settings and concept drift adaptation strategies. Most new malware can survive for 1-5 months under concept drift adaptation strategies, a small portion can survive for more than 5 months, while the survival time of most old malware samples is 0. Moreover, most concept drift adaptation strategies face the situation that some new malware survives for more than one year under sufficient label budget (300 samples). This phenomenon fully indicates that there is still room for improvement in the performance of the current concept drift adaptation strategies. Due to the uneven distribution of survival times for new malware samples under the TRANS concept drift adaptation strategy, the number of samples with survival times of 0-2 months exceeds five times that of samples in other time intervals. Therefore, for the sake of convenient presentation, we have truncated the bar chart representing the 0-2 months survival time samples in the TRANS statistics, while preserving the relative proportions of the overall sample volumes.

D Concept Drift Adaptation Strategy Settings

It is not enough to just enumerate the model structures and concept drift adaptation strategies. What is more critical is to combine them. Our combination follows the principle of optimal configuration. For each target model structure, we select the corresponding concept drift adaptation strategy based on the optimal combination method in existing research methods. The last setting about the concept drift adaptation strategy is the initial state of the target model. Our setting in this study is to train the initial data for 1 year, then conduct CDAD attack verification, and use the monthly time window to evaluate and verify the effectiveness of the attack.

E Concept Drift Adaptation Baseline

We conduct baseline tests on the performance of current mainstream concept drift adaptation methods on the APIGraph [49] and CDAMAL[5] datasets, and the test results are shown in the Figure 8. 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).

Table 13: 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

[5]https://anonymous.4open.science/r/Denial-of-Concept-Drift-Adaptation-Contribution-343C

Here, we offer a concise overview of the hyperparameter configurations. We utilize the SGD optimizer for HCC, TRANS, and UNC while employing the ADAM optimizer for CADE. For the loss functions, TRANS uses the Hierarchical Distance loss function, CADE uses the Triplet Mean Squared Error loss function, and HCC and UNC employe the Hierarchical Distance Cross-Entropy loss function. The total number of training epochs is set to 50 for all four methods. More detailed hyperparameter settings are shown in Table 13. In this study, we test four types of current mainstream excellent Android malware detection models [10, 14, 49], and the obtained concept drift detection data is shown in Figure 8. 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 [10]) demonstrates performance advantage over the existing methods (TRANS [6],CADE [46],UNCER [15]). We notice that the performance of existing concept drift adaptation strategies is unstable, and even the optimal HCC method has performance fluctuations. The instability also inspires the research on the security of concept drift adaptation strategies in our study.

F Freeze Attack Strategy

We observe that the attack value of new high value malware in the initial model cycle also exhibits instability in subsequent model cycles. While 85% of the samples maintain a high attack value in the subsequent cycles, nearly 15% of the samples experience fluctuations in their 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 freeze attacks during the low attack value model cycles of new malware. Freeze attack involves selecting benign attack seeds with the highest uncertainty score. During the poisoned sample generation phase, data augmentation based on sample replication strategies is applied to produce a batch of attack samples.

G 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 strategies 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 strategies, potential ethical concerns are associated with our research. For example, attackers can leverage our methods to carry out attacks or improve malware. Therefore, following previous research precedents [17, 29, 50], we will restrict

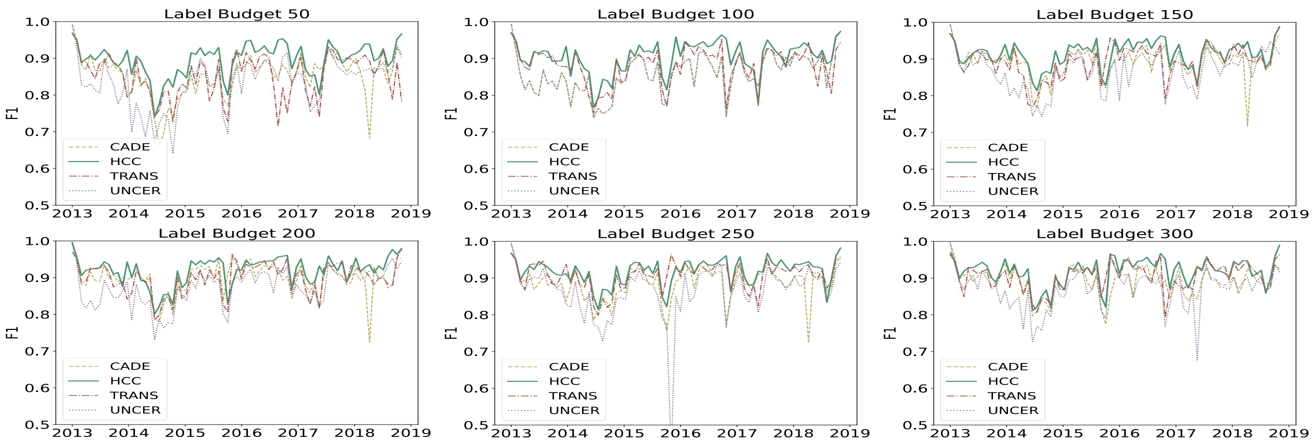

Figure 8: Concept drift adaptation baseline

code sharing to verified academic researchers. In addition, although our CDAMAL dataset is collected through public channels, direct sharing could also provide potential attackers with convenient access for attack testing. Therefore, we will also restrict the sharing of the dataset to verified academic researchers.

H Open Science

In order to enhance the reproducibility and replicability of scientific findings, we will share our research artifacts. Our code (`https://anonymous.4open.science/r/Denial-of-Concept-Drift-Adaptation-Code-C0EF`) and CDAMAL dataset (`https://anonymous.4open.science/r/CDAMAL-Concept-Drift-Dataset-3E08`) are all available. In addition, considering research ethics, our dataset and the source code of the CDAD attack will only share minimized demonstration examples by default to illustrate the attack effectiveness.