CDAD: Concept Drift Adaptation Denial Attack in Android Malware Detection

xxxx xxx
First Institution

xxxx xxx
Second Institution

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 Unfortunately, with the evolution of malware, deployed detection models will soon become outdated, known as concept drift. In order to combat this challenge, various concept drift adaptive methods based on active learning(CDA-AL) are proposed to improve Android malware detection models' performance. The success of CDA-AL is attributed to selecting the most valuable samples during the testing phase and optimizing the model based on these samples. However, we find that CDA-AL exposes a new attack surface, and we call it concept drift adaptive denial (CDAD) attack. In this paper, we conduct the first CDAD attack on Android malware detection models constructed using four mainstream concept drift adaptive strategies. This was performed on an Android malware concept drift dataset containing over 580,000 samples spanning more than 10 years. Experimental results indicate that the latest CDA-AL method is vulnerable to our proposed CDAD attack. Attackers can double the original threat lifecycle of new malware, achieving an average white-box attack success rate of 90% across four mainstream concept drift methods (reaching 80% under the black-box threat model). Furthermore, the impact on the original model's performance during the attack process is less than 2% on average, demonstrating high attack stealth. This means that the protected new malware can spread widely without being detected, posing a serious threat to user mobile security in the real world. Therefore, we aim to draw researchers' attention to the dangers of the CDAD attack.

1 Introduction

Android operating system has become an indispensable part of 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% [43], with nearly 4.74 billion users worldwide [5]. Unfortunately, mobile devices and applications powered by 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 [45].

Facing the massive amount of malware generated daily, it is unsurprising that there is a demand for automatically detecting Android malware [52]. So far, the state-of-the-art ML-based Android malware detectors(ML-AMD) have achieved 99% F-measure in their laboratory settings [26].

However, deploying ML-AMD in the real world faces many significant challenges. One of the most critical challenge is that real world data distribution can change over time, yielding the phenomenon of concept drift [40] [48]. The direct solution is to add new data to the training dataset. However, the number of new Android applications in the real world is overwhelmingly large. Google Play launched 1069 mobile apps everyday in 2024 [9]. Kaspersky detected over 389,000 malware in the first quarter of 2024 [22]. Therefore, it is impossible to obtain sample labels for all new data, which leads to model performance degradation. Some researchers have proved that the F1 value of ML-AMD, which performs well in the training dataset, will drop from 99% to 76% within about 6 months [11]. It can be seen that the concept drift problem is a severe challenge faced by existing machine learning-based Android malware detection models.

Except for adding a large amount of labeled data, the existing optimal methods mainly focus on concept drift adaptation through active learning(CDA-AL) [11]. Recently, several related articles have been published in top venues(Usenix'23 [11],S&P'22 [7],Usenix'21 [50],TIFS'23 [14]). To solve the practical limitations such as label cost mentioned earlier in CDA-AL, the model owner selects only part of the representative data from the test dataset. Then, the model owner manually analyzes this part of the dataset to obtain reliable labels and adds this data to the training dataset for model optimization. But most previous research has primarily focused on improving the performance of these methods. The vulnerabilities of CDA-AL methods have received little attention.

Previous research on the vulnerabilities of Android mal-

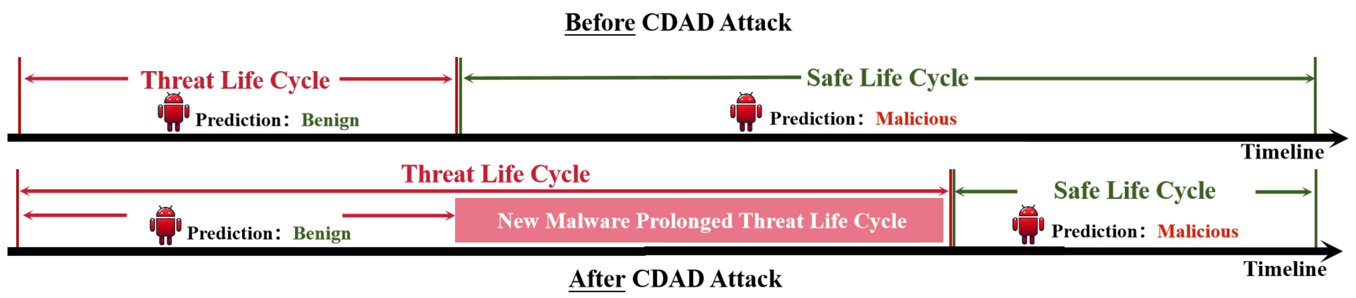

Figure 1: Threat Life Cycle

ware detection models either focuses on the training phase (poisoning attacks [49]) or the inference phase (evasion attacks [4] [25] [33] [39]). And we noted that the concept drift adaptation encompasses not only the model inference and training phases but also the concept drift adaptation phase. Therefore, researching the security of concept drift adaptive methods is of utmost importance. The essence of concept drift adaptation is to deal with the emergence of new malware. So we transform the vulnerability of concept drift adaptation methods into the threat sustainability research of new malware. In other words, we investigate whether there exists an attack method that can compromise the security of concept drift adaptation methods by prolonging the threat life cycle of new malware.

In order to study the sustainability of new malware, we ask three research questions:
Q1-What is the threat life cycle of new malware samples under optimal concept drift adaptation based on active learning?
Q2-Can attackers prolong the threat life cycle of new malware samples in the face of the optimal concept drift adaptive methods?
Q3-How effective is the attack method, and what factors influence the attack's effectiveness?

To the best of our knowledge, the above three research questions are explored for the first time in the field of concept drift adaptation. We explore the answer to the above question by studying the optimal Android malware classifiers with concept drift adaptation capabilities. We choose Android malware because of the availability of public, large-scale, and timestamped datasets that provide malware family information (e.g., AndroZoo [3]).

We answer Q1 by applying recent concept drift adaptation methods [11] against the new Android malware samples. We found that some new malware family samples had a significantly longer threat life cycle than other new malware. For example, the threat life cycle of the tascudap family reached 33 months, which means that samples of this family can survive for more than 2 years. This means that the new malware can carry out operations such as stealing user privacy files and swiping user bank cards for up to 2 years. This phenomenon is not isolated. In every new family each month, there are samples with a threat life cycle greater than 0 in 95% of the families. This implies that there may be certain vulnerabilities in the current active learning methods that can be exploited by attackers, thereby prolonging the threat life cycle of new malware. For detailed data on the threat life cycle of new malware, refer to Figure 9 in Appendix B.

To answer (R2), we propose the Concept Drift Adaptive Denial (CDAD) attack to extend the threat life cycle of new malware. Our proposed method can efficiently generate targeted poisoning poison samples with clean labels using existing real world malware, without the need for generative models.

To address Q3, we conducted an evaluation of the attack's effectiveness and analyzed the factors influencing the attack. Regarding attack effectiveness, we evaluated it under both white-box and black-box threat models, and analyzed the factors influencing the attacks. Finally, we analyzed the attacker's costs to illustrate the severe threat our proposed attack method poses to real-world mobile security.

Evaluation and Insights To better demonstrate the vulnerability of mainstream CDA-AL techniques in the field of concept drift adaptation 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) [50], and uncertainty(UNCER) [16]. Our attack validation dataset spans 10 years, 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 98% against new malware. Moreover, the threat life cycle of over 85% of new malware can be prolonged by 100%.

To summary, Our key contributions are as follows:

- We have discovered significant security vulnerabilities in the concept drift adaptive models proposed in recent top security conferences.The threat life cycle of new malware can be effectively extended by 100% by attackers. 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 Concept Drift Adaptive Denial (CDAD) attacks. This framework can efficiently generate attack samples with clean labels, and it does not require any modifications

to the malicious samples that need protection. This is beneficial for attackers in spreading malware.

- We conducted attack effectiveness tests on a dataset of hundreds of thousands of real Android software over a period of 10 years, and conducted detailed discussions on the influencing factors of attack effectiveness, as well as attack concealment, and the costs of attackers and defenders. We found that the proposed attack method can achieve an average attack success rate of over 84.02%, with extremely strong concealment(F1 change less than 1.8% before attack), and the attacker 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 for ML-AMD

Different CDA-AL methods have differences in concept drift adaptive strategies, but they also share common behavior patterns. An illustration of CDA-AL methods is shown in Figure 2.

2.1.1 Behavior pattern of CDA-AL

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 through the model interface. The target model gives a prediction label. In the case of binary classification, the return label value is 0 or 1.

Stage2-Sample selection: The primary goal of selecting samples for a test dataset is to address the issue of how to allocate labeling budgets for new test data when the overall labeling budget is limited. The 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: In this stage, 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.

2.1.2 Concept Drift Adaptive Strategy

Currently, the core problem of CDA-AL is finding high-value concept drift samples in the sample selection process, that is concept drift adaptive strategy. The mainstream CDA-AL methods in the field of Android malware concept drift adaptation include the following four concept drift adaptation strategies.

1) Model Uncertainty The core idea of Uncertainty measurement [16] is to detect concept drift based on the output layer of the target model. In the process of concept drift adaptation, the model will give priority to selecting the samples with high uncertainty of the current model for labeling and adding them to the training dataset to update the model. 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 [50] introduced the encoder spatial distance for concept drift detection. Specifically, CADE 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 [?] introduced the thery of conformal prediction [46](credibility and confidence) into the field of concept drift. Given a new test sample, they first compute the non-conformity score of the sample, representing how dissimilar it is from the training set. Given the predicted label of the test sample, they find the set of calibration data points with the same ground truth label. Then, they compute credibility as the percentage of samples in the calibration set that have higher non-conformity scores than the test sample. They compute 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.(Usenix'23 [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, and the model is trained end-to-end based on this loss.

2.2 Adversarial Attacks on ML-AMD

Adversarial machine learning attacks are being widely studied in multiple fields [8] and pose a significant threat to the large-scale deployment of machine learning solutions in security-critical environments. Currently, adversarial attacks against ML-AMD can be roughly divided into two categories: evasion attacks and poisoning attacks.

Evasion Attacks have been widely explored in computer vision, and previous research has also investigated the applicability of such techniques in malware classification [10, 27, 38, 54]. 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

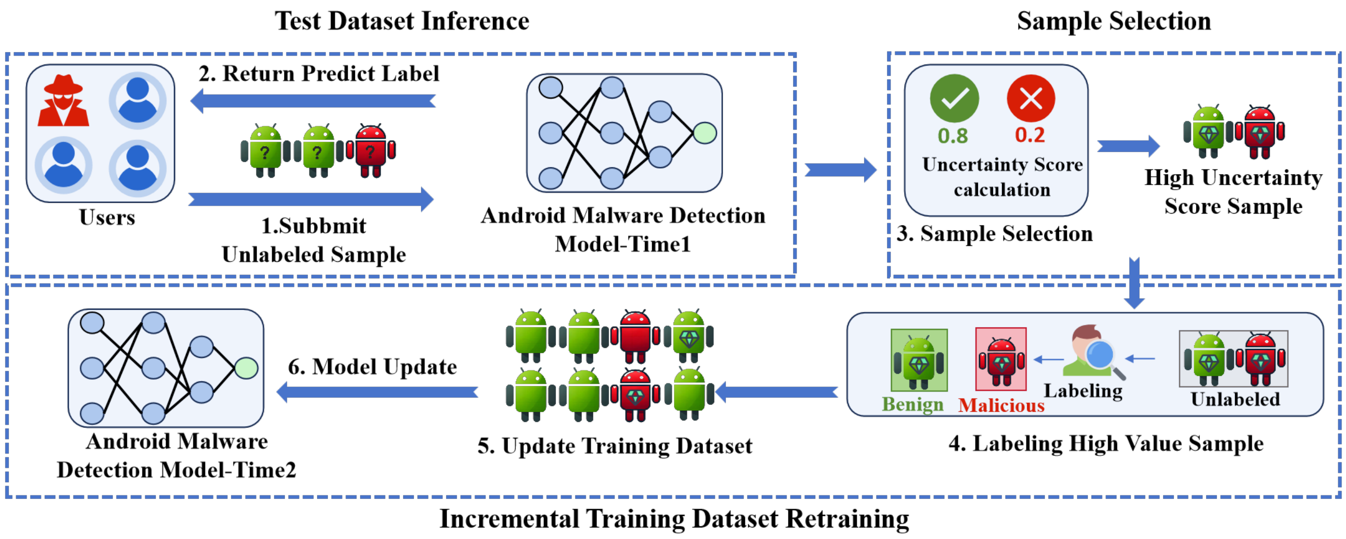

Figure 2: Concept Drift Adaptation Based Active learning

Table 1: Adversarial Attack on AMD-ML

Venues	Category	Attack Phase
TIFS'19 [10]	Evasion	Inference
CCS'21 [54]	Evasion	Inference
Usenix'21 [41]	Backdoor	Inference
TDSC'21 [26]	Backdoor	Training
CCS'23 [18]	Evasion	Inference
S&P'23 [49]	Backdoor	Training
Our Work (CDAD)	Poisoning	Training + Inference

an adversarial example.

Poisoning Attacks are one of the most dangerous threats to machine learning models [26, 41]. These attacks assume attackers can inject poisoned samples into the training dataset at will. Due to the large scale of the training datasets, it is difficult to manually detect poisoned samples. 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 two categories - 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 [26, 28, 41] 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 training process of the model (poisoning attacks), as shown in Table 1. However, we found that in addition to the training phase, poisoning attacks are also very likely to occur during the concept drift adaptation phase. In the subsequent sections of this paper, we focus on the impact of poisoning attacks during the concept drift adaptation process.

3 Motivation

Our motivation comes from a key real world observation: the longer the threat life cycle 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 [51], 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. With a huge number of user infections, every extra day of new malware survival means attackers get a lot of money. In addition, it should be noted that the top 10 mobile security threats in 2023 [60] are all positively correlated with the threat life cycle, which means that the longer the threat life cycle, the greater the harm of these threats.

In order to understand the extension of the threat life cycle of new Android malware under the existing optimal detection methods, we use the current optimal concept drift adaptive Android malware detection model [11] to quantitatively analyze the threat life cycle of emerging malware samples.

Figure 3 shows the changes in various model performance indicators of the latest existing concept drift adaptation methods [11] from 2012 to 2018. As can be seen from figure 3, nearly 15.44% of malware samples will have false negative detection results every month, which means that these malware samples can survive under the current model and then perform operations such as privacy theft or malicious payment. These data fully demonstrate that the risk of prolonged threat life cycle of new malware is not an isolated case, but exists in large numbers.

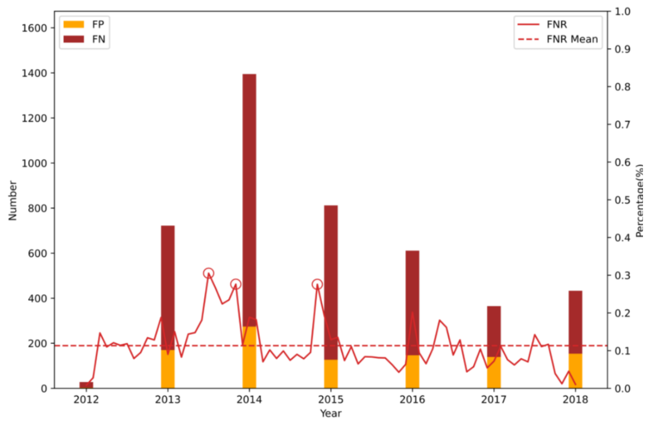

Figure 3: Model Performance Changes Over 6 Years under the Optimal Concept Drift Adaptation Method

Specifically, as shown in Figure 3, some new malware samples still have a long threat life cycle under the current optimal concept drift adaptive method. Among them, families such as ‘gomanag’ can even survive for more than 5 years(62 months) under the current optimal concept drift adaptive method. Additionally, we analyzed the optimal concept drift adaptive methods under multiple labeling budget settings and found that new families with a threat life cycle greater than one month accounted for an average of 73.75% of all new families in that month. Therefore, addressing the increasingly prolonged threat life cycle of new malware is crucial.

Research Motivation: Unfortunately, there is a lack of research on how attackers can achieve threat life cycle enhancement under concept drift adaptation. Therefore, we investigate the risk of new malware threat life cycle enhancement in the most powerful existing concept drift adaptive detection 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.

4 Threat Model

The attacker’s goal is to mislead the model’s decision boundary by submitting poison samples, thereby enhancing the threat life cycle of the new malware to be protected. To ensure that the research hypothesis is consistent with the real world situation of the enhanced life cycle of new malware threats. 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 system’s active learning incremental training dataset, concept drift adaptation strategies, sample feature vectors, and model parameters. This setting follows Kerckhos’ principle [34], ensuring that the security of the model does not rely on secrecy. This assumption is realistic since CDA-AL methods should be publicly available so that they can be rigorously vetted for security before deployment like cryptanalysis. Additionally, considering that CDA-AL may ultimately favor some state-of-the-art methods [12], it will be challenging to maintain strict confidentiality. We further validated the effectiveness of our proposed attack scheme under the conditions of a black-box threat model. In a black-box threat model [59], 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 [44, 57].

5 Attack Methodology

5.1 Adversary’s Challenges

Although attackers have some capabilities and knowledge(mentioned in Section 4), attacks against CDA-AL still face severe challenges, these challenges make our attack significantly different from previous attacks. The detailed challenges faced by attackers are as follows:

Attack Sustainability: Previous poisoning attacks [28] 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 sustainability 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 sustainability of poisoning attacks on the active learning process for Android malware.

Feature perturbation does not change the sample label: This paper focuses on test-time concept drift adaptive attacks in feature space. The challenge is to give adversarial feature vectors without affecting the original sample labels of the problem space objects. In other words, after perturbations in the feature space, the original samples still retain their original labels in the problem space.

Training Process is out of control: In our threat model, the attacker cannot directly poison the training dataset, nor can he control the training process of the target model. They only have sample submission and query access to the latest state of the target CDA-AL model. For example, compared with attack scheme proposed by Giorgio et al.(Usenix’21 [41]), 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 enhancing the threat life cycle. However, this method requires attackers to pay more code development costs and is difficult to use frequently, because frequent

Table 2: Adverisal-Challenge(●:have this characteristic,○:lack of this characteristic)

Attack Method	Attack Sustainability	Training out of control	Malware Integrity	Label correctness	Attack Concealment
TIFS'19 [10]	○	○	○	●	●
CCS'21 [54]	○	○	○	●	●
Usenix'21 [41]	○	○	○	●	●
TDSC'21 [26]	○	○	○	●	●
CCS'23 [18]	○	○	○	●	●
S&P'23 [49]	○	○	○	●	●
Our Work(CDAD)	●	●	●	●	●

updates may alert detectors, and rewriting malware means that the victim users who have been controlled may be lost. Therefore, we consider maintaining malware integrity as 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 [55]. 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 Overview

Taking the above attack challenge as the prerequisite, we propose a Concept Drift Adaptation Denial(CDAD) attack method, which consists of three steps: surrogate model training, malware attack value assessment, and malware threat life cycle extension. The overall workflow is shown in figure 4.

The first step is to conduct a surrogate model, it is responsible for simulating the target model, providing a basis for subsequent attack steps (see Section 5.2.1 for details). The second step is to conduct malware attack value assessment. This module is responsible for assessing the attack value of new malware samples and selecting the appropriate attack strategy(see Section 5.2.2 for details). The final step involves the generation of poisoned samples through the Malware Threat Life cycle Extension Module.(see Section 5.2.3 for details).

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. Because once new malware is provided to the target model, the target model may perform manual analysis on the new malware, which will increase the cost of the attack. In addition, subsequent attack operations need to rely on the understanding of the target model to evaluate the attack value of new malware and improve attack efficiency(see Section 5.2.2 for details).

The training process of the surrogate model is independent of the training process of the target model. The surrogate model collects new data samples from the real world, identifies concept drift samples, and obtains query results for concept drift samples based on the target model. 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. Refer to Appendix A for further details. Then the surrogate model is retrained based on concept drift samples to ensure that its detection performance is always close to the target model. It is worth noting that the purpose of the attacker is to approximate the detection ability of the target model, so he does not need to care about the true label of the query sample, but only needs to get the sample prediction result (pseudo label) of the target model, which greatly reduces the label cost of the attacker. For example, with a concept drift adaptive method that has a labeling budget of 200, an attacker can save the labeling analysis cost for 2400 Android samples annually.

In addition, the surrogate model construction method in this paper is different from the previous shadow model construction methods. Because the CDA-AL 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 time nodes. We define the interval between the two model update time nodes as a unit model period t_i. In the unit model period , the model parameters of the target model θ_{di} and the surrogate model θ_{ai} remain stable. Therefore, after each update of the target model, the attacker queries the target model θ_{di} for the pseudo labels $\hat{Y}_i$ of the latest concept drift samples D_{ci}, and updates the surrogate model with this samples. By repeating the above process, the attacker will continue to obtain a series of surrogate model($\theta_{a0}, \theta_{a1}, ..., \theta_{an}$) with detection capabilities similar to the series of target models($\theta_{d0}, \theta_{d1}, ..., \theta_{dn}$). Each 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. The concept drift detection function f_{cd} involved in Algorithm 1 corresponds to the four concept drift adaptive methods mentioned in Section 2.1.

5.2.2 Malware Attack Value Assessment

Based on the surrogate model, attackers are able to assess the attack value of new samples. Through this assessment, attackers can reduce attack costs, improve attack efficiency, and provide different attack strategies for different new malware.

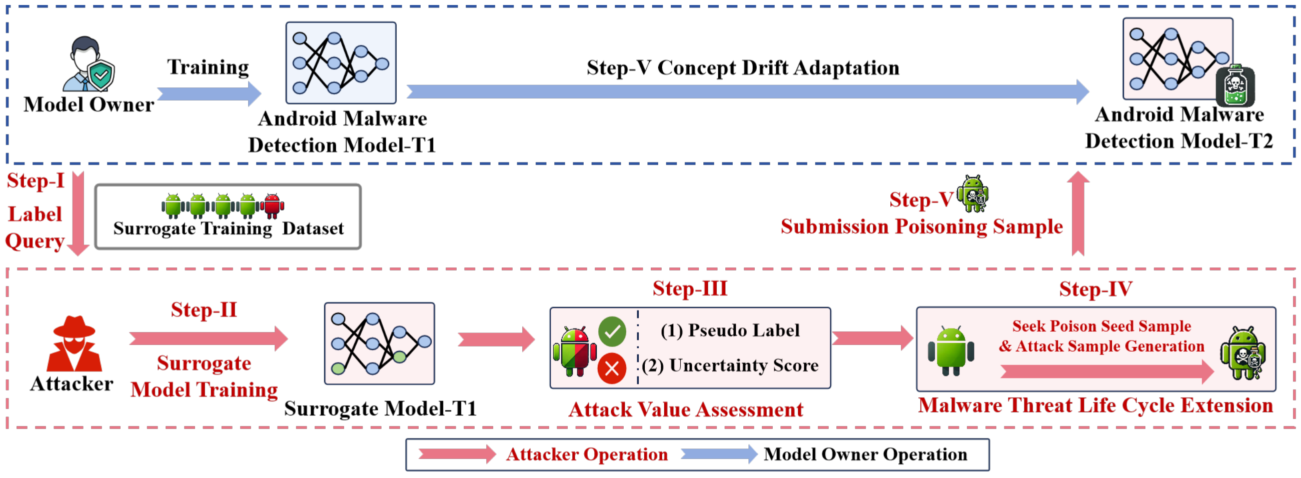
Model Owner
Training
Android Malware Detection Model-T1
Step-V Concept Drift Adaptation
Android Malware Detection Model-T2
Step-I Label Query
Surrogate Training Dataset
Step-V Submission Poisoning Sample
Attacker
Step-II Surrogate Model Training
Surrogate Model-T1
Step-III
(1) Pseudo Label
(2) Uncertainty Score
Attack Value Assessment
Step-IV
Seek Poison Seed Sample & Attack Sample Generation
Malware Threat Life Cycle Extension
Attacker Operation
Model Owner Operation

Figure 4: Concept Drift Adaptation Denial Attack

Algorithm 1 Surrogate model training

```
Input: initial model $\theta_0$, new testing samples appearing within unit model period $D_0, D_1, D_2, .....D_i$, target models in different model periods $\theta_{d0}, \theta_{d1}....., \theta_{dn}$, active learning label budget $LB$, loss function $L$ for model training process
Output: surrogate model sequence $\theta_{a_0}, \theta_{a_1}....., \theta_{a_n}$
1:  for each $t_i$ $i \in [0,n]$ do
2:      Step I: Calculate Uncertainty Scores
3:      calculate $D_i$ uncertainty scores $U_i$   ▷ As defined in 2.1
            $U_i \leftarrow f_{cd}(D_i)$
4:      Step II: Sample Selection
5:      get concept drift samples $D_{ci}$ (within the label budget $LB$ ) based on the uncertainty score $U_i$
            $D_{ci} \leftarrow Select(D_i, Count)$
6:      Step III: Label Query
7:      query the target model $D_i$ for the pseudo labels $\hat{Y}_i$ of concept drift samples $D_{ci}$
            $\hat{Y}_i \leftarrow Query(D_{ci}, LB)$
8:      Step IV: Model Retraining
9:      add concept drift sample data $D_{ci}$ to the training dataset $D_T$
            $D_T \leftarrow UpdateDataset(D_T, D_{ci})$
10:     retrain the surrogate model $\theta_{ai}$ based on the training dataset $D_T$ to obtain a new surrogate model $\theta_{a(i+1)}$
            $\theta_{a(i+1)} \leftarrow Training(D_T, L, \theta_{ai})$
11: end for
12: return $\theta_{a0}, \theta_{a1}....., \theta_{an}$
```

In the absence of a attack value assessment method, attackers have to adopt a unified attack strategy for all new types of malware, resulting in a substantial increase in attack costs. Additionally, the success rate of the attack will also decrease, as illustrated in Section 6.3.

The malware attack value assessment is specifically divided into two sub-modules: T threat capability assessment and T+1 threat capability assessment. Algorithm 2 shows the process of malware attack value assessment.

(1)T attack value assessment: The T threat capability assessment 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 be protected to the surrogate model θ_{di} for evaluation. If the evaluation result of the surrogate model is false negative, it is considered that the new malware samples have a high probability of survival in the current model cycle.

It should be noted here that the sample cannot be submitted to the target model θ_{ai} for evaluation. Because once the evaluation result of the target model to the sample is positive, the target model θ_{ai} will add the malware sample x_m to its own blacklist, which will help to improve the performance of the target model θ_{ai} and increase the difficulty of subsequent attack operations. If the false negative judgment constraint is not met, the attack value of the current samples x_m is low. For example, the attacker simply uses a simple repackaging method or slightly modifies the sample in an attempt to avoid hash matching of the target model.

(2) $T+1$ attack value assessment: Although T attack value assessment can eliminate samples with no attack value, it cannot prioritize the samples with attack value. Therefore, we propose the $T+1$ attack value assessment.$T+1$ threat assessment can confirm the attack priority of samples with attack value, and cooperate with T threat assessment.

Specifically, $T+1$ threat assessment takes the survival probability in the next model cycle as the basis for judging the attack priority of emerging malware. Considering the characteristics of active learning process, the survival probability of emerging malware in the next model cycle is usually affected by sample selection caused by concept drift adaptive strategy. Therefore, when the uncertainty score of new malware is lower than the lowest score for sample selection in the current model cycle, the new malware will not be subjected to manual analysis, resulting in a disruption of the threat cycle.

This type of new malware has a higher attack value among the new malware identified in the T threat assessment.

Algorithm 2 Malware Attack Value Assessment

```
Input: surrogate model for a specific model cycle $\theta_{ai}$, the
    minimum test score for sample selection in the specific
    model cycle $min_i$, new testing samples appearing within
    unit model period $D_{t0}, D_{t1}, D_{t2}, .....D_{ti}$ new testing sam-
    ples appear in the current model cycle $D_{ti}$,Malware need
    to protected $x_m$
Output: attack value of malware need to be protected $v_m$
 1: Step I: T Attack Value Assessment
 2: evaluating new malware to be protected $x_m$ based on
    surrogate model $\theta_{ai}$
        $\hat{y}_m = ModelPredict(\theta_{ai}, x_m)$
 3: T Attack value assessment based on surrogate model
    predictions $y_m$
 4: if $y_m == 1$ then
 5:     $v_m = low$
 6: else
 7:     $v_m = high$
 8: end if
 9: Step II: T+1 Attack Value Assessment        ▷ Only
    high-value samples in the T attack value assessment can
    enter the second step T+1 attack value assessment
10: calculate uncertainty score of new malware to be pro-
    tected $x_m$ based on surrogate model $\theta_{ai}$
        $u_{x_m} \leftarrow f_{cd}(x_m)$
11: if $u_{x_m} < min_i$ then
12:     $v_m = high$
13: else
14:     $v_m = middle$
15: end if
16: return $v_m$
```

It is worth noting that the $T+1$ threat assessment represents the sample attack priority setting under limited attack resources, but the samples with different scores all have attack value. Specifically, if the T threat capability assessment passes, but the $T+1$ threat assessment fails, it is a medium attack value sample. Attackers will employ the freezing attack mentioned in Section 5.2.3.2 to perform low-cost attack operations.

To further illustrate the role of the attack value assessment module, we conducted necessity analysis experiments. This part of the experiment verifies the important role of attack value assessment in reducing the cost of attack. For more detailed data see Experimental Evaluation Section 6.3.

5.2.3 Malware Threat Life Cycle Extension

Next, we enter the new malware threat life cycle extension module, which is the core of the CDA-AL attack framework. The main function of this part is to improve the threat life cycle of high-value new malware samples given by the attack value assessment module. Malware threat life cycle extension mainly includes three parts, namely poison seed sample selection, poison sample generation and poison sample data distribution disturbance.

5.2.3.1 Poison Seed Sample Selection

$$U_i = f_{cd}(x_i), \quad x_i \in (D_T \cup D_{ti}) \wedge y_i = 0 \tag{1}$$

$$X_s = \{x_i | U_i(x_i) > \tau\} \tag{2}$$

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 attacker, as shown in Equation 1 and 2. Searching for high-uncertainty samples to use as seeds for generating attack samples ensures the influence on the target model's decision boundary, thereby extending the TLC of new malware. It should be noted here that the seed samples of the poisoning attack are benign samples, which provide the basis for subsequent clean label attacks. We define the entire data available to the attacker (including the existing training dataset and the test dataset acquired that month) as the search space for identifying poisoned attack seed samples. Due to the characteristics of concept drift, the uncertainty scores of samples in the newly acquired test dataset are typically higher than those in the existing training dataset. Therefore, although including the test dataset in the search space incurs additional labeling costs, it provides a richer selection of potential attack seeds. Through the above operations, we obtained the poison sample seeds X_{seed} that enhance the threat life cycle of new malware samples to be protected.

5.2.3.2 Poison Sample Generation The search for seed samples in poisoning attacks ensures that the samples obtained by the attacker 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, it is challenging to meet the required capacity for poisoning attack samples. The range of sample selection in active learning typically involves a set of samples rather than a single sample. Moreover, the stronger the capability of the target model, the better its sample labeling capacity, resulting in a larger range for sample selection.

Especially considering that the manual analysis budget is set in the mainstream CDA-AL (for example, Chen et al.(Usenix'23 [11]) set label budget as 50, 100, 200, 300), too few poisoned samples are difficult to have a big impact on the CDA-AL method. Therefore, the challenge faced by attackers is how to generate a large number of effective poison samples based on a small number of 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. 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 feature space search to confirm the feature space of samples that meet the attack requirements. The second part feature space inverse mapping is to illustrate 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.(SP'20 [35]), so our attack approach primarily focuses on constructing the feature space for poisoning samples.

Feature Space Search: 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 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 malicious samples, thereby enhancing the model's detection capability for such 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. Otherwise, considering the characteristics of active learning, the poison samples will not be able to 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 protected new malware 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 uncertainty score constraints. In data disturbance augmentation, the attacker disturbs the original feature space of the seed sample. The specific steps are shown in Algorithm 3.

It should be noted here that the feature_perturbation function in Algorithm 3-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. 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 the constraint

Algorithm 3 High Attack Value Malware Threat Life Cycle Extension

Input: poisoned seed samples X_s, original feature space dimension M of samples, and the surrogate model θ_{ai}, Minimum sample uncertainty score for the current model period min_i, uncertainty score for new protected malware u_m.

Output: A collection of poison sample data that meets the constraints D_p.

```
1:  Step I: Feature space perturbation
2:  while keeping the label stable, perform feature perturbation on sample $X_s$, form a series of perturbed datasets $D_{sk}$.
3:      for each $k \in len(X_s)$ do
4:          ▷ constraint 1 $D_{sk} = feature\_perturbation(x_{sk})$
5:      end for
6:      $p = len(X_s) - 1$
7:      $D_s = [D_{s0} \cup D_{s1} \cup .....D_{sp}]$
8:  Step II: Uncertainty Score Evaluation
9:  The generated samples are submitted to the surrogate model $\theta_{ai}$ for testing, and the surrogate model feed back the uncertainty score of the perturbed datasets $D_s$.
10:     for each $j \in len(D_s)$ do
11:         $u_{x_{sj}} = f_{cd}(x_{sj}, \theta_{ai})$
12:         if $u_{x_{sj}} < min_i$                ▷ constraint 2 then
13:             $D_s = D_s - x_{sj}$
14:         end if
15:         if $u_{x_{sj}} < u_m$                  ▷ constraint 3 then
16:             $D_s = D_s - x_{sj}$
17:         end if
18:     end for
19:     $D_p = D_s$
20:     return $D_p$
```

1. Under the premise of satisfying constraint 1, the attacker performs uncertainty filtering on the generated samples to ensure that the obtained sample uncertainty score is greater than the lower limit of the label budget(constraint 2) and at the same time greater than the uncertainty score of the sample to be protected(constraint 3). 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 that protected malware samples 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 samples to be protected.

Feature Space Inverse Mapping: In the end, we need to show that the features corresponding to the poison samples must have corresponding Android applications in the problem space. Since the attack seed sample is an object in the problem space that meets semantic requirements, and the poison samples are single-bit flip variants of the attack seed sample in the feature space, so they also meet semantic requirements in the problem space. For feature spaces such as the APIGraph dataset and the Abdrozoo dataset, a single transformation means an increase or decrease in an API call or permission declaration.

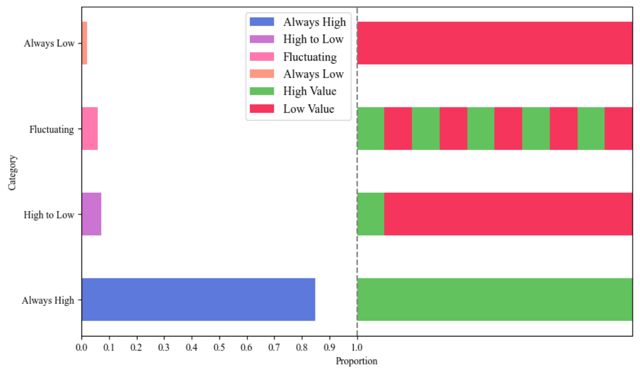

Figure 5: Changes in attack value

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 threat life cycle extension module need to be integrated in each model cycle. Furthermore, the attack value of high-value new malware in the initial model cycle is also unstable in subsequent model cycles, as shown in Figure 5. 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 attack 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.

5.2.3.3 Poison sample data distribution disturbance The above attack method can prolong the TLC 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. 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 malicious samples.

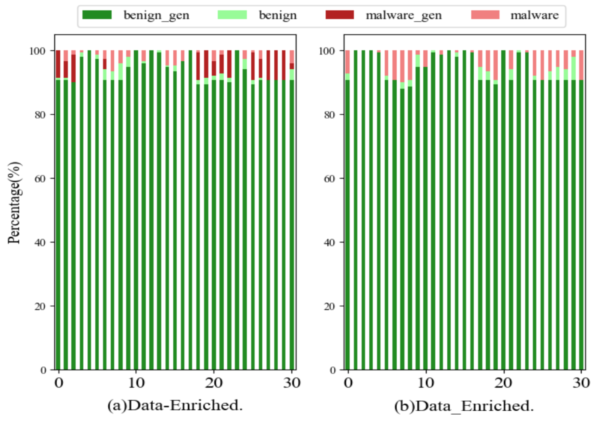

Figure 6: Poison Sample Data Distribution Disturbance

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 extension of the TLC 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 as an addition 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 3. As shown in Figure 6, we conducted statistics on the data distribution of attack samples. Figure 6-a shows the data distribution of concept drift samples after introducing the perturbation module for attack sample data distribution, while Figure 6-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.

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 [23, 36], etc., we can do more in-depth research on malware evolution characteristics. In the process of constructing a concept drift dataset, data collectors are required to engage in extensive and long-term collection of Android software. Additionally, the dataset must provide comprehensive information on various malware families. 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 [53] and BJTU-AMCD) to validate the effectiveness of our attack methods. Both datasets are characterized by extensive, long-term collection and rich family information. The second dataset was collected and constructed by us. Details on the collection and construction process can be found in Appendix A. Additionally, the two datasets we selected focus on permission features and API features, respectively. This allows us to effectively validate the robustness of our attack methods under different feature construction approaches.

Table 3: Evaluation Dataset

Year	Malware	Benign	Total	New Family
2012	3061	27472	30533	45
2013	4854	43714	48568	34
2014	5809	52676	58485	26
2015	5508	51944	57452	15
2016	5324	50712	56036	49
2017	2465	24847	27312	42
2018	3783	38146	41929	55
2017	2108	18972	21080	29
2018	4625	41625	46250	23
2019	8612	77508	86120	36
2020	3512	31608	35120	35
2021	4903	44127	49030	25
2022	2814	25326	28140	38
Total	57378	528677	586055	452

The combined time span of both datasets covers 11 years. The number of samples in the dataset reaches 580,000, with more than 700 malware families. Detailed information is shown in Table 3.

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 adaptive strategies for concept drift.

For the target model setup, we based our approach on the current best practices in the field of concept drift(Usenix'23 [11]) and enriched it by model mentioned in He et al.(CCS'23 [18]) and Fang et al.(TIFS'23 [14]). The specific model setting combination is shown in Table 4. In setting up the concept drift adaptation strategies, we tested not only the current best strategies(Usenix'23 [11]) but also several key milestone approaches(SP'22 [7],Usenix'21 [50]).

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 adaptive 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 adaptive attack verification, and use the monthly time window to evaluate and verify the effectiveness of the attack. The final combination is shown in Table 4.

Table 4: Combination of Model Structure and CDA-Strategy

Venues	Strategy	Structure
Usenix'23 [11]	HCC	Encoder-Classifier(MLP) [11]
SP'22 [7]	TRANS	Classifier(SVM) [6]
Usenix'21 [50]	CADE	Encoder-Classifier(MLP) [31]
[16]	UNCER	Classifier(ResNet) [14]

6.1.3 Surrogate Model Settings

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 Section 4. Weak attackers, lacking knowledge of the model's structure, parameters, and training dataset, conduct attacks under the black-box threat assumption. This settings is used to evaluate the effectiveness of attacks when the attacker's capabilities are limited. A detailed analysis will be provided in Section 6.2.3.

6.1.4 Evaluation Matrix

In terms of performance indicators, we use F1, Precision, Accuracy, and other indicators to measure the overall performance of the model perfomence. Regarding attack effectiveness metrics, we constructed the Attack Success Rate (ASR) based on the threat life cycle. The definition of a successful attack includes two criteria. The first criterion is that the attacker can prolong the threat life cycle of new malware by a factor of two, which we refer to as a complete attack success. The second criterion is that the attacker prolongs the threat life cycle of the new malware, but not by a factor of two. Considering the characteristics of concept drift adaptive attacks and the profit model of new malware, such attack outcomes can still result in significant losses to users' mobile security. Therefore, we also refer to this as attack success.

6.1.5 Attack Objective Settings

Attack objective dataset settings: To verify the effectiveness of the attack, we selected samples from a new family of malware as the target for testing. If the new family contains multiple samples, one sample will be selected for attack verification. It is worth noting that if the FNR index of the new family is 1, such a family has higher attack value than those new families with an FNR not equal to 1. Therefore, we select samples from such new families for attack 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 threat life cycles(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 7. 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.

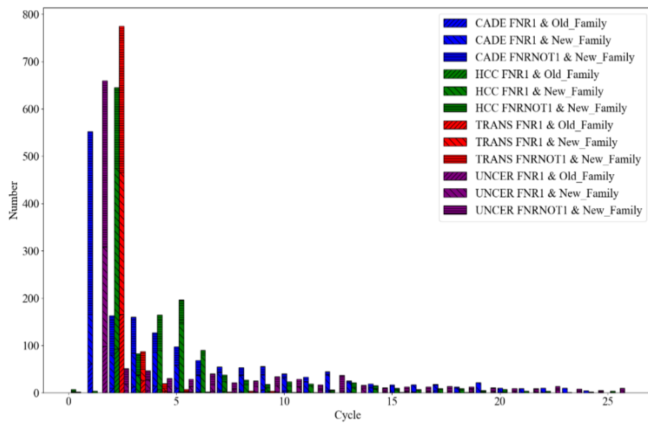

Figure 7: Original Threat Life Cycle

Attacker attack cycle settings: In theory, attackers can perform persistent enhancements to the threat life cycle 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 threat life cycle.

6.2 Attack Effectiveness

Based on the settings of the above dataset, target model, surrogate model and evaluation matrix, we start to evaluate the effectiveness of the 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 attacker have white-box access to model information. On this basis, we introduce the attack algorithm process to carry out concept drift adaptive attack on the above malware to prolong its threat life cycle. We conducted attack tests on 4 models and the corresponding concept drift adaptation strategies, where each group of test objects tested 6 different label budgets (50, 100, 150, 200, 250, 300).The detailed test data configuration is shown in Table 5.

First, we observed that the poison samples significantly degrade the target model's ability to detect new malware. Through extensive experimental 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 TLC 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. Through extensive experimental evaluation, it can be observed that the F1 performance indicator varies within 1.8%. Even under the UNCER setting, the F1 performance has a slight increase compared with active learning, indicating the concealment of the 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, thereby prolonging the threat's life cycle. 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 threat life cycle by double, 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.

After fully confirming the effectiveness of the attack method, we conducted tests on the attack time cost. We evalu-

Table 5: Verification of the effectiveness of concept drift adaptive attack

Concept Drift Strategy	F1(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	ASR(%)
HCC(Latest)	$90.25_{(-1.76)}$	$85.88_{-2.80}$	$0.44_{+0.04}$	$99.56_{-0.04}$	$14.12_{\mathbf{+2.80}}$	**89.44**
TRANS	$89.39_{-0.59}$	$85.18_{-1.12}$	$0.53_{\mathbf{-0.01}}$	$99.47_{\mathbf{+0.01}}$	$14.82_{\mathbf{+1.12}}$	**79.07**
CADE	$88.13_{-1.74}$	$85.57_{-2.62}$	$0.90_{+0.06}$	$99.10_{-0.06}$	$14.43_{\mathbf{+2.62}}$	**84.72**
UNCER	$88.05_{\mathbf{+0.61}}$	$83.86_{\mathbf{+0.98}}$	$0.65_{+0.00}$	$99.35_{\mathbf{+0.00}}$	$16.14_{\mathbf{-0.98}}$	**82.86**

ated data from 2013 to 2018 and found that the current optimal concept drift adaptive 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 adaptive methods, which update models on a monthly basis. For specific configuration parameters of the test environment for attack time cost, refer to the Appendix B.

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 and that the approach is effective over a concept drift lasting up to 6 years. 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 search space size, label budget and the proportion of the label budget of the label budget.

Table 6: Search Space Influence Factor

Search Space	F1(%)	FPR(%)	FNR(%)	ASR(%)
100	90.25	0.44	14.12	**89.44**
90	89.00	0.22	14.54	**87.93**
80	87.44	0.38	19.16	**91.28**

(1) Search Space Size: The sample feature search space refers to the entire feature space that the attacker can perturb when generating perturbed samples based on the attack seed samples. In the attack scheme of this paper, the perturbation feature space formed after a single bit flip is performed on each attack seed sample is the entire search space of the 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 and sample evaluation. In this 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 adaptive method as the attack target, and set the label budget to 200. The experimental evaluation results are shown in Table 6.

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 and cannot traverse the entire feature search space, they can still carry out effective attacks.

(2) Label budget variance: Label budget is an important parameter in the concept drift adaptation process and 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 adaptive attack, we set up multiple sets of comparative experiments for analysis. We do not consider the possibility that the attacker's label budget is higher than the defender's, because in reality defenders are often large organized security companies, while attackers are usually individuals or small organizations. Under the setting of label budget alignment, the experimental data refers to Table 7. It can be seen that no matter how much the label budget is, the attack is effective, and we can see that as the label budget increases, the attack effect increases by 2%. Especially the currently optimal solution(Usenix'23 [11]), the main reason is that this method is an active learning method based on the loss function. Therefore, samples within the sample selection range have a stronger impact on the target model. As the label budget increases, the attacker's ability to influence the target model becomes stronger.

(3) The proportion of label budget occupied by the label budget: 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

Table 7: Attack Effectiveness under Different Label budget

Strategy	Budget	F1(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	ASR(%)
HCC	**50**	**87.67**$_{-2.00}$	81.25	0.38	99.62	18.75	**81.56**
HCC	**100**	**89.75**$_{-0.73}$	84.66	0.37	99.63	15.34	**84.77**
HCC	**150**	**90.32**$_{-1.09}$	85.53	0.36	99.64	14.47	**85.99**
HCC	**200**	**90.25**$_{-1.76}$	85.88	0.44	99.56	14.12	**89.44**
HCC	**250**	**91.33**$_{-0.46}$	86.82	0.32	99.68	13.18	**89.71**
HCC	**300**	**89.74**$_{-2.02}$	84.31	0.35	99.65	15.69	**89.33**

Table 8: proportion of label budget occupied by the label budget

Proportion	F1(%)	FPR(%)	FNR(%)	ASR(%)
100	90.25	0.44	14.12	**89.44**
70	90.15	0.43	14.22	**86.67**
50	90.04	0.46	14.21	**81.52**

set the label budget proportions to 100%, 95%, 90%, 85%, and 80%. We then evaluated how different label budget proportions affect the attack outcome. As shown in Table X, 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.

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. A weak attacker is unable to access the parameters and training data of the target model. Since a weak attacker needs to carry out a separate training process, they will incur additional computational costs.

Specifically, compared to the strong attacker setting, we made the following adjustments. In terms of mastery of target model information, the structural information of the surrogate model is no longer aligned with the target model. Since complex models represent greater computational consumption, we primarily considered model settings under weakened attacker capabilities. 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, corresponding to a weakened encoder, a weakened classifier, a simultaneous weakening of both the encoder and classifier, and an original control group.

Previous research has indicated that clean-label attacks can suffer from end-to-end performance degradation [47], as model updates can lead to a deterioration in attack effectiveness. However, the CDAD attack method presented in this paper alleviates the issue of diminished attack effectiveness under end-to-end conditions. The experimental result data(Table 9) shows that the average ASR under various black-

Table 9: Attack Effectiveness Under Model Heterogeneity

Model Heterogeneity	F1(%)	FNR(%)	ASR(%)	R-ASR(%)
Equal-Enc&Cla	89.46	15.20	**77.27**	**82.95**
Weak-Enc	86.62	17.25	**72.31**	**59.09**
Weak-Cla	83.60	21.38	**82.22**	**44.31**
Weak-Enc&Cla	86.37	17.93	**87.18**	**39.77**

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 achieves an impact on the target model's updates, enhancing the attack 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.

Additionally, we have observed an interesting phenomenon. The ASR under the synchronized weakening of both the encoder and classifier is 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 not only the ASR but also the number of attack samples.

6.3 Attack Value Assessment necessity analysis

Table 10: Attack Value Assessment necessity analysis

Setting	Ablation-ASR(%)	ASR (%)	Improvement (%)	Attacker Budget(%)
HCC	**81.73**	**86.21**	**↑4.48**	**↑16.97**
CADE	**72.43**	**88.85**	**↑16.42**	**↑18.51**
UNCER	**61.50**	**66.90**	**↑5.40**	**↑19.52**

Analysis of the attack value of new malware samples provides support for subsequent threat life cycle 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 16 different concept drift strategies and label budgets, the absence of the attack value assessment module led to an average decrease of 20.44% in attack success rates. Notably, the attack success rate of the UNCER-150 strategy group showed the most significant drop, declining by 74.63%. 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 10. This demonstrates that the attack value assessment module is an essential part of CDAD attacks.

6.4 Attack Explainability

This section further analyzes and explains why the attack is effective from the perspective of the model decision boundary. We use six representative new malware families or variants that appeared in 2013 as examples to explain the effectiveness of attacks. In the original active learning process, we can see that with the addition of high-uncertainty scoring samples, the model decision boundary gradually changes, and the new malware samples gradually move closer to the decision boundary from being far away from the decision boundary, and are finally detected as malware. But with the introduction of concept drift adaptive attack, we can see that the model decision boundary enters a state of slow updating. New malware family samples are always far away from the decision boundary, which leads to the continuous expansion of the threat life cycle. The reason for the above phenomenon is that the change of the decision boundary depends on high-uncertainty samples. Although our poisoning poison samples have high uncertainty and include both benign and malware samples, the contribution of poisoning samples to the update of the decision boundary is very small. Malicious samples come from samples that have been learned by the original model, while the new samples appear to be a group of high-uncertainty samples, but in essence they come from a poisoned attack seed sample.

6.5 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 proxy 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 C. 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 [18].

7 Discussion and Future Work

In this study, we introduce an efficient framework for Concept Drift Adaptive Denial (CDAD) attacks. Our work aims to offer valuable insights into the security of the AMD-ML field and raise awareness regarding the potential threats posed by the life cycle of new malware. In the following, we discuss some limitations of our approach 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

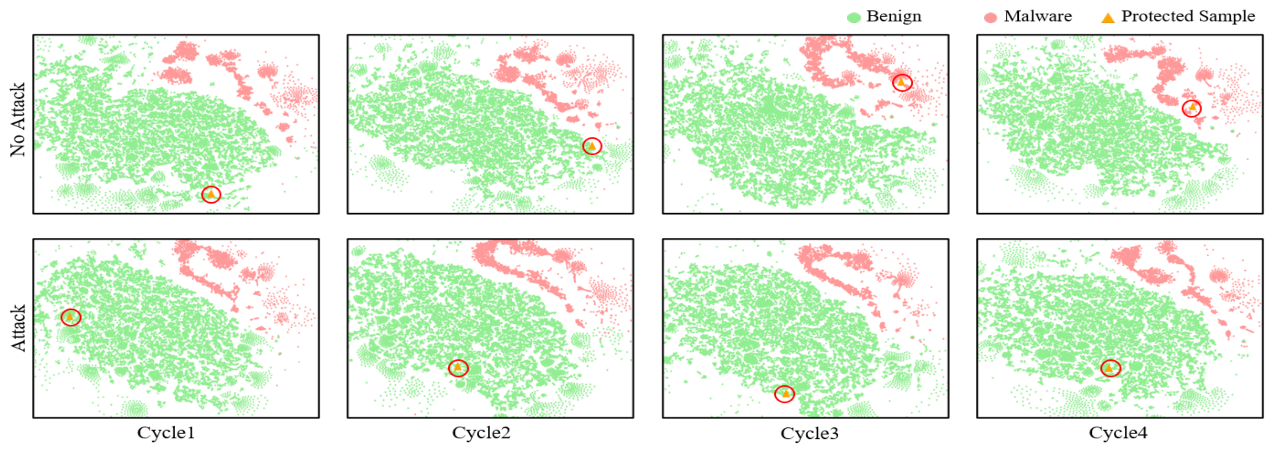

Figure 8: Attack Explainability

success rate or stealthiness of the attack. In future research, we plan to tackle these specific cases.

Defense Strategy: As we know, the current adaptive methods for concept drift based on active learning lack corresponding research on defense strategies. Only Lin et al. [29] have conducted research on defense strategies against poisoning attacks in active learning specifically targeting Android malware. However, the current method assumes that the attacker has the ability to maliciously mislabel, while the scheme in this paper is based on the assumption of clean-label attacks. Furthermore, this defense method relies on a clean data subset that is not accessible to the attacker for validation, which is impractical in scenarios where data acquisition for Android malware is open. 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 based on the existing uncertainty assessment to form a defensive screening strategy for concept drift samples. 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. Additionally, the defense method needs to ensure continuous resistance against attacks during the multiple rounds of iterative updates in the model's active learning process, while minimizing defense overhead as much as possible.

8 Related Work

This work is broadly related to works on threat life cycle of Android malware,concept drift and poisoning attack.

Malware Threat Life Cycle. Research on the threat life cycle of malware primarily focuses on the dynamic changes of malware detection results and their influencing factors over time. Shuofei Zhu et al. [58] have been the first to observe fluctuations in the threat life cycle 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. However, this work attributes the fluctuations in the threat life cycle of malware to the detection quality of the detection engines, but lacks research on the reasons behind the differences in detection quality among these engines. Yun Shen et al. [42]have conducted studies on the threat life cycle from another perspective and found that even the threat life cycle of malware is not only influenced by detection results, but also by the actual detection program permissions. They find that while app markets remove potentially harmful apps after these become known, there is a significant delay between when potentially harmful apps are identified and when they are removed: potentially harmful apps persist on Google Play for 77 days on average and 34 days on third party marketplaces.Inspired by the large-scale data analysis and statistics in Work 1, our research focuses on the influencing factors of the threat life cycle of malware.

Concept Drift. Currently, researchers believe that the phenomenon of concept drift is the main reason affecting the threat life cycle of malware. Existing research can be classified into three major categories.1)concept drift detection and adaptation, researchers have proposed a series of methods [7, 11, 21, 50] to detect concept drift samples and optimize models based on such samples. The scheme proposed by Yizheng Chen et al. [11]is currently the latest research achievement. 2)learning robust features, researchers are seeking more robust sample feature extraction methods to cope with concept drift [53].3)explanatory against Concept Drift, Researchers have recently embarked on studying the interpretability of concept drift, with the aim of providing guidance for effective drift adaptation. Yiling He et al. [19]and their team have proposed DREAM, a method that can effectively enhance the accuracy of drift detection and reduce the cost of manual analysis. Our work primarily focuses on the detection of concept drift, as this phase serves as the foundation for subsequent stages. While existing research has mainly concen-

trated on optimizing the performance of adaptive methods for concept drift, our emphasis lies on its security. This is a very natural idea, as the proposal of adaptive methods for concept drift aims to mitigate the problem of fluctuating threat life cycles posed by malware. Therefore, we investigate whether adaptive methods for concept drift can be influenced by attackers, thereby controlling changes in the threat life cycle.
Poisoning Attack. Poisoning attacks are one of the primary methods for compromising target models. Existing poisoning attacks against deep neural networks primarily fall into four categories [47].1)Gradient-based method perturbs the training data in the direction of the adversarial objective function's gradient, until the poisoned data achieves maximum impact [20, 30].2)The key to generative method lies in training a generative model that produces a large number of poisoned samples by adding adversarial perturbations [15, 32].3)Influence-based method quantifies the impact of poisoned data on model predictions, enabling the creation of the most effective poison [13, 24].4)Clean-label method introduces subtle perturbations to the training data to create poisoned samples while leaving the data labels unchanged [17, 56]. Since the growth of the threat life cycle of malware is a pattern of attack service, it is crucial to ensure the integrity of the malware. Inspired by generative method and clean-label method, we propose a poisoning attack method that generates a large number of poisoned samples without altering the data labels.

9 Conclusion

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

Acknowledgments

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] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pages 3019–3025, 2020.

[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] Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. Learning to confuse: Generating training time adversarial data with auto-encoder. *Advances in Neural Information Processing Systems*, 32, 2019.

[16] 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.

[17] 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. *arXiv preprint arXiv:2009.02276*, 2020.

[18] 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.

[19] Yiling He, Junchi Lei, Zhan Qin, and Kui Ren. Going proactive and explanatory against malware concept drift. *arXiv preprint arXiv:2405.04095*, 2024.

[20] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems*, 33:12080–12091, 2020.

[21] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*, pages 625–642, 2017.

[22] Kaspersky. It threat evolution in q1 2024. mobile statistics. https://securelist.com/it-threat-evolution-q1-2024-mobile-statistics/112750/, 2024.

[23] 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.

[24] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, pages 1–47, 2022.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

[31] 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.

[32] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

[41] 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.

[42] Yun Shen, Pierre-Antoine Vervier, and Gianluca Stringhini. A large-scale temporal measurement of android malicious apps: Persistence, migration, and lessons learned. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1167–1184, 2022.

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

[44] 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.

[45] 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.

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

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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?

[60] ZIMPERIUM. Global-mobile-threat-report. https://www.zimperium.com/global-mobile-threat-report/, 2023.

A Collection of Concept Drift Datasets

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 by Rahali et al. [37], resulting in the final Android malware concept drift dataset. For detailed feature information, please refer to Table 11.

Table 11: 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		

B CDA-AL Baseline

Table 12: Parameter setting of active learning method

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

We conducted baseline tests on the performance of current mainstream concept drift adaptation methods on the APIGraph [53] 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 12.

C Testing of attack overhead in problem space

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

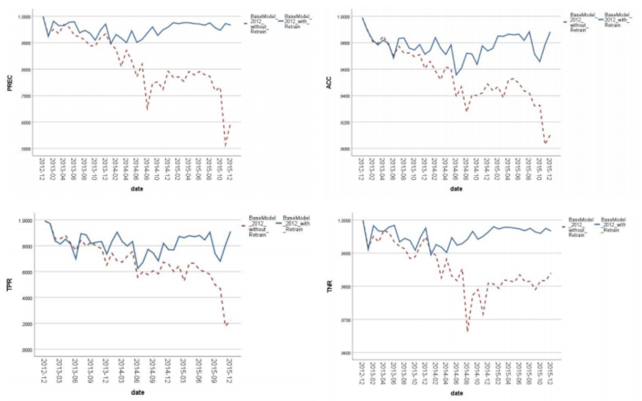

Figure 9: CDA-AL Baseline

quickly map attack samples from the feature space to the problem space. We selected APKs of different types and sizes,

Table 13: APK obfuscation time

APK	Size(MB)	Category	Obfuscation
JD	97.59	shopping	54.95s
Taobao	57.03	shopping	78.98s
Little Red Book	162.99	social	178.68s
Google	315.67	tool	93.32s
Wang VPN	45.51	tool	14.91s
WeChat	264.04	communication	136.76s
Average	199.72	-	90.72s

and tested their corresponding repackaging and obfuscation time, as shown in Table 13.

D Potential Ethical Concerns

The main purpose of our research is to evaluate the security of CDA-AL methods, as CDA-AL related methods have currently received attention from researchers. Attackers are motivated to exploit the vulnerabilities of CDA-AL methods to prolong the threat life cycle of new malware, thereby gaining more benefits. Even though the intent is strict about evaluating the robustness of CDA-AL 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 [18, 35, 54], we will restrict code sharing to verified academic researchers.