⓪ Add Tomal to this paper.

CDAD: Concept Drift Adaptation Denial Attack in Android Malware Detection

Anonymous submission

Abstract

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 Unfortunately, with the evolution of malware, deployed detection models will soon become outdated, a phenomenon known as concept drift.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

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

1 Introduction

① too strong

The Android operating system has become indispensable to people's lives over the last decade. As of January 2024, the Android operating system ranked first in the global operating system market share, reaching 41.63% [37], with nearly 4 billion active users worldwide [12]. Unfortunately, mobile devices and applications powered by the Android operating system have been selected as valuable targets by cyber criminals [5]. According to a security analysis by Kaspersky, even the official Google Play Store had over 600 million malware downloads in 2023 [41]. Facing the massive amount of malware generated daily, researchers have proposed automatic detection methods for Android malware based on machine learning [48].

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

② on average?

③ to address what problem

Existing solutions [6, 10, 15, 46] mainly focus on concept drift adaptation through active learning for Android malware detection. ~~In order~~ To tackle the practical limitations previously mentioned, such as the cost of labeling, the aforementioned research [6,46] has devised diverse evaluation methods aimed at assessing the value of test data. Then researchers introduce high-value samples to mitigate model performance degradation caused by concept drift. But most previous research [6, 10, 15, 46] focus on improving the performance of concept drift adaptation. The vulnerabilities of concept drift adaptation through active learning for Android malware detectors have received little attention.

⑤ I don't know what the problem is.

⑥ a new concept but I don't know the meaning.

Previous research on the vulnerabilities of Android malware detection either focuses on the security of training phase (poisoning attacks [13, 21, 27, 34, 45]) or the security of inference phase (evasion attacks [4, 20, 29, 32]). And we note that the concept drift adaptation focuses on the concept drift samples selection, which is different from the training and

④ What do you mean by this? So only high-value samples are labelled?

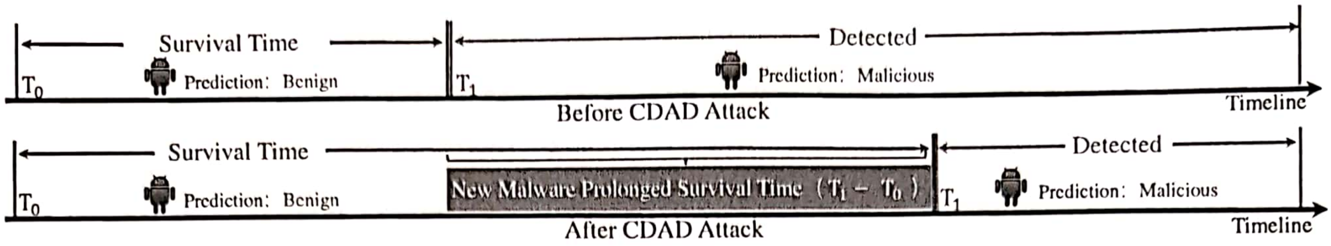

Figure 1: Prolonged survival time of new malware under CDAD attack

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

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

What ??

③ What is 0 month?

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

To better demonstrate the vulnerability of existing concept drift adaptation methods to CDAD attacks, we conduct experiments on four existing Android malware concept drift detection strategies, including Hierarchical Contrastive Classifier (HCC) [10], Transcending (TRANS) [6], high-dimensional outlier distance (CADE) [46], and uncertainty (UNCER) [15]. Our attack evaluation dataset spans 10 years and the total number of samples of our dataset reaches more than 580,000. The experimental evaluation results show that our CDAD attack method achieves an attack success rate of 92.77% on the latest Strategy (HCC).

④ Complete the sentence.

To sum up, we mainly make the following contributions:

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

2 Background

2.1 Concept Drift Adaptation Paradigm

Different concept drift adaptation strategies share same paradigm (as shown in Figure 2). The classical paradigm consists of three stages. Stage I is test dataset inference. New Android applications are submitted to the detection model for testing. The detection model gives a predicted label. In the case of binary classification, the return label value ~~is 0 or 1.~~ ⑤ Stage II is concept drift sample selection. The primary goal of sample selection is to find concept drift samples. It can address the issue of how to allocate labeling budgets for new data when the overall labeling budget is limited. Researchers design different sample selection strategies to find the most helpful samples for improving the model's performance under limited label budget. Stage III is incremental training dataset retraining. The model trainer adds selected samples as an incremental update part of the training dataset. After retraining,

② What does this even mean?
another new concept. why do we have to create a name for it?

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

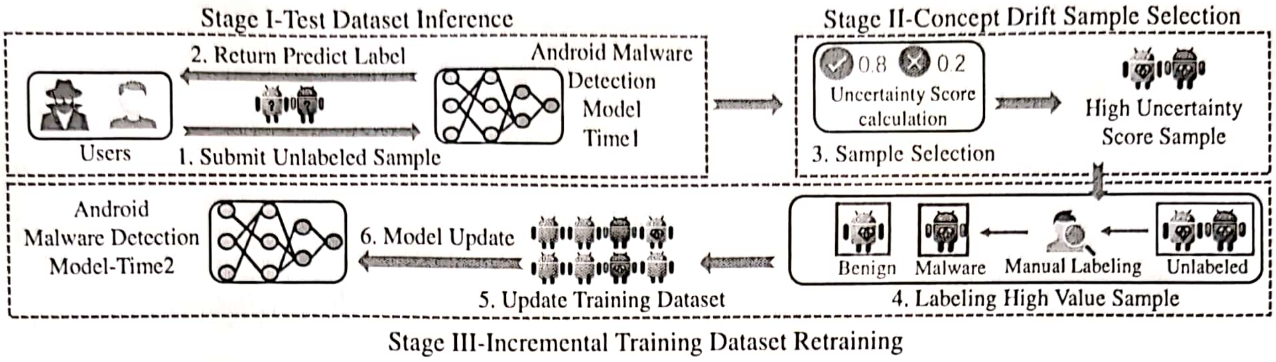
Stage I-Test Dataset Inference
2. Return Predict Label
Android Malware Detection Model Time1
Users
1. Submit Unlabeled Sample
Stage II-Concept Drift Sample Selection
0.8
0.2
Uncertainty Score calculation
High Uncertainty Score Sample
3. Sample Selection
Android Malware Detection Model-Time2
6. Model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

Figure 2: Concept drift adaptation based active learning

model performance degradation caused by concept drift can be alleviated.

① are these sample selection strategy?

2.2 Concept Drift Adaptation Strategy

The existing concept drift adaptation strategies in the field of Android malware detectors can be classified into the following four strategies.

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

Encoder Space Distance. CADE [46] trains an encoder through existing labeled data for learning a compressed representation (dimension reduction) of a given input distribution. Then, the newly obtained test samples can be provided to the encoder to obtain the encoder's spatial features. Finally, the distance function can effectively identify concept drift samples far from the existing training dataset.

② did any paper use this "term"?

Credibility and Confidence. Transcending [6] introduced the thery of conformal prediction [42] (credibility and confidence) into the field of concept drift adaptation. Given a new test sample, Transcending first computes the non-conformity score of the sample. Then, it computes credibility as the percentage of samples in the calibration set that have higher non-conformity scores than the test sample. Finally, it computes confidence as one minus the credibility of the opposite label. A lower credibility score or a lower confidence score means the test sample is more likely to have drifted.

Hierarchical Contrastive Loss. The method proposed by Chen et al. [10] is currently the best-performing strategy in Android malware concept drift adaptation methods. The model consists of two modules. The first module is an encoder and the second module acts as the classifier. In terms of loss function settings, to ensure that the model is robust to concept drift, the training loss of the model is set to the sum of hierarchical contrast loss and classification loss. The advantage of this strategy is its provision of finer-grained encoder rules and utilization of similar features among malware families, ultimately enhancing concept drift adaptation performance.

The names are not even consistent with Intro. (HCL)?

2.3 Attacks on Android Malware Detection

Currently, adversarial attacks against Android malware detectors can be roughly divided into two categories: evasion attacks and poisoning attacks.

Evasion Attacks have received extensive attention in the field of Android malware detection [9,22,31,50]. Specifically, the attacker's goal in an evasion attack is to add a small perturbation to a target malware sample to get it misclassified. Such perturbed example is called an adversarial example.

③ Why are we talking about this? Is this paper about Evasison attacks?

Poisoning Attacks are one of the most dangerous threats to machine learning models [21,34]. These attacks assume attackers can inject poisoned samples into the training dataset. In poisoning attacks, the adversary's goal is to degrade model performance through some malware modifications to the training dataset. After being trained on the poisoned dataset, the model's performance degrades at test time. According to the different degradation degrees of the victim model, poisoning attacks can be roughly divided into untargeted and targeted poisoning attacks. The goal of untargeted poisoning attacks is to decline the overall performance of the victim model. The goal of targeted poisoning attacks is to force the victim model to perform abnormally on a specific input class. Backdoor attacks [13,21,34] are a special case of targeted poisoning attacks where the victim model only misclassify samples containing specific triggers.

④ What?

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

This doesn't read like Background but related work.

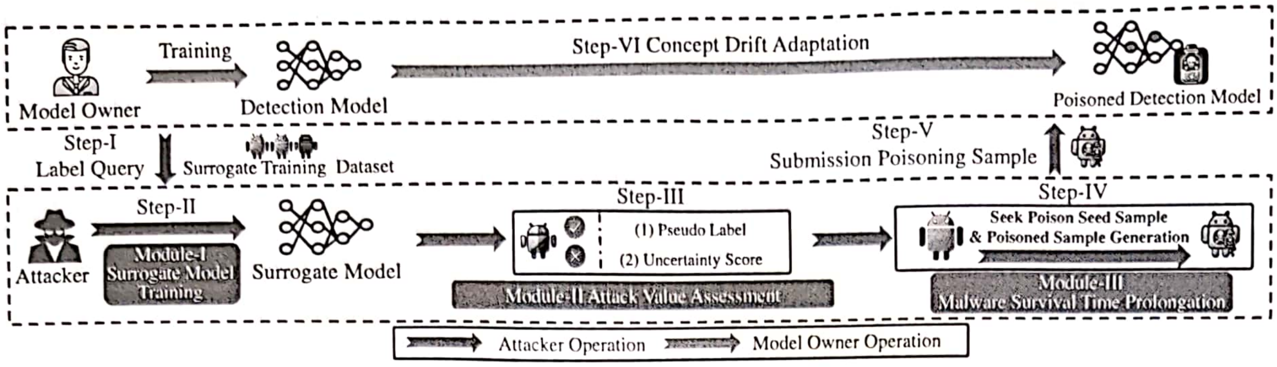
Model Owner
Training
Detection Model
Step-VI Concept Drift Adaptation
Poisoned Detection Model
Step-I
Label Query
Surrogate Training Dataset
Step-V
Submission Poisoning Sample
Step-II
Attacker
Module-I Surrogate Model Training
Surrogate Model
Step-III
(1) Pseudo Label
(2) Uncertainty Score
Module-II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

Figure 3: Concept drift adaptation denial (CDAD) attack

poisoning

I assume this is a concept/term coined by others

3 Attack Methodology

3.1 Threat Model

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

① Do you really need all of these?? This is too strong.

??? ②

3.2 Adversary's Challenges

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

Attack Continuity (AC). Previous poisoning attacks [18, 25, 40, 47] typically validate their effectiveness under the condition of a fixed training dataset without considering the scenario where the training dataset is continuously updated. One of the most notable features of active learning is that it continually introduces new data into the training dataset. Therefore, ensuring the continuity of the attack's effectiveness during the model update process is challenging. This paper is the first to study the continuity of poisoning attacks on the concept drift adaptation process for Android detection models.

③ I don't get what you are saying.

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

④ What ???

I don't understand the point of this whole subsection.

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

⑤ Why is this important?

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

3.3 Attack Method

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

⑥ Is this an approach used in other papers?

3.3.1 Surrogate Model Training

We build a surrogate model for obtaining the information needed for subsequent attack operations without providing

① That is a big assuption.

new malware to the target model. The training process of the surrogate ~~model is independent~~ of the training process of the target model.

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

??

② What the heck is this?

However, as attacker, we lack the ability to provide reliable labels for concept drift samples. We collect new data samples D_t^i from the real world, identifies concept drift samples $\bar{D_c^i}$, and obtains query results as pseudo labels for concept drift samples $\bar{D_c^i}$ based on the target model θ_d^i. It is important to emphasize that due to the openness of the Android platform, our ability to collect data aligns with the owner of target model, so we all get Android samples D_t^i. Additionally, our purpose is to approximate the detection ability of the target model θ_d^i, so we do not need to care about the true label of the concept drift samples $\bar{D_c^i}$ but only needs to get the sample prediction result (pseudo label) of the target model θ_d^i, which greatly reduces our label cost. Then, the surrogate model θ_a^i is retrained based on concept drift samples $\bar{D_c^i}$ to ensure that its detection performance is always close to the target model.

In addition, our surrogate model construction method differs from the previous shadow model construction methods. Because the target model has the ability to update dynamically, our surrogate model must also be constantly updated. We define the time interval between the two model update time nodes as a unit model period. In a unit model period (at time node i), the model parameters of the target model θ_d^i and the surrogate model θ_a^i remain stable. Therefore, after each update of the target model, we query the target model θ_d^i for the pseudo labels $\hat{Y}_i$ of the latest concept drift samples $\bar{D_c^i}$ and updates the surrogate model θ_a^i with these samples. By repeating the above process, we continue to obtain a series of surrogate models $(\theta_a^0, \theta_a^1, ..\theta_a^i...)$ with detection capabilities similar to the series of target models $(\theta_d^0, \theta_d^1, ..\theta_d^i...)$. Each surrogate model object in the model sequence represents the approximation of the detection ability of the surrogate model to the target model.

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

Algorithm 1: Surrogate model training

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

3.3.2 Malware Attack Value Assessment

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

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

2) attack value assessment based on uncertainty score: