Poster: Towards Adversarial Detection of Mobile Malware

Sen Chen[†], Minhui Xue[†‡], Lihua Xu[†]
[†]East China Normal University, Shanghai, China
[‡]NYU Shanghai, Shanghai, China
Email: {ecnuchensen, minhuixue}@gmail.com; lhxu@cs.ecnu.edu.cn

ABSTRACT

Android malware has been found on various third-party online markets, which poses drastic threats to mobile users in terms of security and privacy. Machine learning is one of the promising approaches to discriminate the malicious applications from the benign ones. Despite its higher malware detection capability, a significant challenge remains: in adversarial environment, an attacker can adapt by maximally sabotaging classifiers by polluting training data. This paper proposes *KuafuDet*, a two-phase learning enhancing approach that adversarially detects the Android malware. Experiments on more than 50,000 Android applications demonstrate the effectiveness and scalability of our approach.

1. INTRODUCTION

There has been a plethora of research in malware detection for Android, wherein machine learning is one of the promising techniques [2]. Machine learning approaches also have a weakness: they are susceptible to adversarial countermeasures by attackers aware of their use. First, through reverse-engineering, attackers may become aware of classifiers and their parameters used to evade detection. Second, more sophisticated attackers can actively tamper with the classifiers by injecting the well-crafted data into training data. Therefore, with Android's policy of open-source kernel, malware writers can gain an in-depth understanding of the mobile platform, hence intentionally alter the training set to reduce or eliminate its detection efficacy.

In this paper, we consider a threat model within a specific class of attacks, named poisoning attacks, in which the attacker is assumed to control a subset of samples or inject additional seeds at will in order to mislead the learning algorithm. We also assume the attacker has full access to the classifier used, and can inject as many variants' features as possible at will to the given classifier. To test the ramifications of causative attacks, we incrementally develop an adversarial model with three types of attackers and then propose *KuafuDet*, a learning enhancing system with effective self-adaptive learning. *KuafuDet* includes an offline training phase wherein selects and extracts contributing features from the training set, and an online detection phase wherein utilizes the classifier trained by the first phase. Compared to the state of the art, these two phases act together, through a self-adaptive learning scheme, as an iterative adversarial detection process. In addition, we intuitively introduce the camouflage detection for verifying false alarms to protect against poisoning attacks.

To the best of our knowledge, our work is the first to detect Android malware in adversarial environment by designing a two-phase machine-learning detection system.

MobiCom'16 October 03-07, 2016, New York City, NY, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4226-1/16/10.
DOI: http://dx.doi.org/10.1145/2973750.2985246

2. OVERALL ARCHITECTURE

The overall architecture of *KuafuDet* is shown in Figure 1, which is comprised of two intertwined phases: In the **Training Model** phase, *KuafuDet* extracts features from labelled application based on our combined set of contributed features and trains classifier model offline; In the **Online Detection** phase, *KuafuDet* classifies large sets of online Android applications into different categories, benign and malicious; Meanwhile, *KuafuDet*, through a **Self-adaptive Learning** scheme, discovers new information from both the identified malware and the filtered suspicious false negatives from **Camouflage Detector**, and incorporates into **Training** to stabilize the detection accuracy.

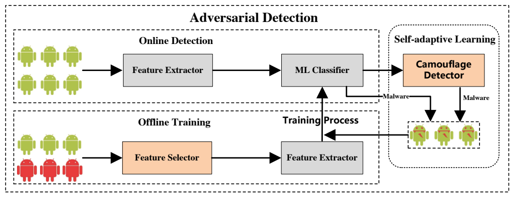

Figure 1: The *KuafuDet* Framework through Adversarial Detection

2.1 Implementation

In order to perform accurate and scalable adversarial detection, our proposed adversarial detection approach contains two phases, training and detection, intertwined by the self-adaptive learning (SAL) scheme. The implementation of *KuafuDet* involves the following steps:

(i) In the feature selection stage, we decompile APKs to generate Smali code via Apktool,[1] we extract 224 out of 564 features using manual pruning along with information gain validated, as partially shown in Table 1.

(ii) In the training stage, we use different machine learning classifiers, such as Support Vector Machines (SVM), Random

[1]http://ibotpeaches.github.io/Apktool/

Table 1: Features

Syntax Features				Semantic Features
Permission	**Intent**	**Hardware**	**API Call**	**Sequence**
READ_PHONE_STATE	INTENT.CATEGORY.VIEW	HARDWARE.TELEPHONE	URL.getContent	(“chmod 777”, getRuntime, exec, “su”)
SEND_SMS	INTENT.CATEGORY.HOME	HARDWARE.CAMERA	Runtime.getRuntime	(“phone”, getSimSerialNumber, URL, openConnection)
INSTALL_PACKAGES	INTENT.ACTION.SEND	HARDWARE.TOUCHSCREEN	SmsManager.getDefault	(“wifi”, getConnectionInfo)
……	……	……	……	……

Table 2: The performance of adversarial detection

Conventional Detection	SVM			RF			KNN		
FN	4.90%			2.50%			3.40%		
Accuracy	94.95%			96.35%			95.80%		
Attacker	Weak	Strong	Sophisticated	Weak	Strong	Sophisticated	Weak	Strong	Sophisticated
Without AD	SVM			RF			KNN		
FN	8.60%	49.80%	62.60%	5.60%	41.80%	55.90%	5.90%	26.20%	45.40%
Accuracy	93.10%	72.50%	64.30%	94.80%	76.40%	67.85%	94.55%	84.40%	72.00%
Within AD	SVM			RF			KNN		
FN	5.80%	11.00%	16.40%	4.70%	15.80%	19.50%	4.10%	10.20%	15.40%
Accuracy	94.50%	91.90%	87.40%	95.65%	89.70%	86.35%	95.45%	92.40%	88.55%

Forests (RF), and K-Nearest Neighbors (KNN), based on 224 dimensional features we selected.

(iii) In the camouflage detection stage, we perform similarity-based filtering to identify the false negatives that are the camouflaged malicious applications.

2.2 Camouflage Detector

To further discover camouflage in malware, we manually pick a fair number of applications from the farthest very benign outcomes and very malicious outcomes from the classification hyperplane, respectively. Those hand-picked applications are the most benign and most malicious predictions. We then use Jaccard index (J_T) and Cosine similarity (C_T) to measure the similarity of applications. If the similarity between two applications exceeds a certain threshold, the application will be selected as a malware candidate and fed back to the training process for further fine-grained detection. We want to select as many malware candidates as possible for periodically retraining the classifiers. To be specific, a low threshold likely leads to high false negatives, while a high threshold leads to low false negatives. During our experiments, we empirically choose the $0.55 < J_T < 0.70$ and $0.35 < C_T < 0.55$ as the thresholds for picking the camouflage malware.

3. EXPERIMENTAL EVALUATION

We evaluate *KuafuDet* on more than 50,000 applications downloaded from various popular third-party Android markets, as well as from real industrial platforms, such as Pwnzen Infotech Inc.

(i) Evaluation on attacks against the detection. In adversarial machine learning (AML) attacks [3], attackers may make great efforts to have a direct influence by corrupting the training set. Here, we analyze the robustness of our discriminative classifiers when encountering three distinct attack strategies (see Table 2). The first attack strategy is to launch a causative attack without any knowledge of the training data or ground truth. This *weak attacker* in principle amounts to injecting noise into the system. The second attack strategy corresponds to the *strong attacker*, who only manipulates partial features in the training set. The third, the most aggressive attacker we consider is the *sophisticated attacker*. This attacker can fully manipulate almost all training features to launch a sophisticated attack, which creates scenarios where relatively benign mobile applications and real-world malicious mobile applications appear to have nearly identical attributes at the training stage.

The weak attacker is not able to force the accuracy of our malware detection below 90%. This suggests that discriminative classifiers can be relatively robust to this type of random noise-based attack. When dealing with the strong attacker, performance degrades to approximately 90% accuracy. The sophisticated attacker can cause the accuracy to drop to approximately 65% by incorporating thousands of training set. The sophisticated attacker represents a practical upper bound for the accuracy loss that a realistic attacker can inflict on our detection system. We see that injecting carefully crafted data into training data can significantly reduce detection accuracy.

With the help of adversarial detection, holistic performance upgrades by at least 15% accuracy with respect to each listed classifier. Analysis on false negatives has an analogous interpretation.

(ii) Evaluation on accuracy. Our accuracy rate (96.35%) completely outperforms the accuracy rate in StormDroid (93.80%) [2] and DREBIN (93.90%) [1]. We achieve the highest accuracy because of the feature selection and similarity-based filtering.

(iii) Evaluation on time cost and scalability. Average detection time per application is less than 3 seconds, which is indeed capable of scaling up to the massive data sets.

4. CONCLUSION

In this paper, we showed how the conventional machine learning classifiers can fail against determined attackers. Based on these insights, we designed and evaluated three types of attackers targeting the training phases to poison our detection. Through simulation, we presented practical bounds for the accuracy loss to each attacker. To address this threat, we therefore proposed our detection system, *KuafuDet*, and showed it significantly reduces false negatives and boosts the detection accuracy by at least 15%.

Acknowledgements

This work was supported in part by the NSFC (No. 61502170, No. 61361136002), and in part by the Science and Technology Commission of Shanghai Municipality (No. 13ZR1413000).

5. REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*, 2014.

[2] S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu. Stormdroid: A streaminglized machine learning-based system for detecting android malware. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 377–388. ACM, 2016.

[3] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.