Attacks on DroidAPIMiner

Ilan Gold
315804583
Computer Science, Ariel University, Ariel, Israel

Abstract

In this Article we will talk about DroidApiMiner: we will examine the classifier and present the weak spots that we found the attack we did on the classifier and see how did it effect the classification process. The attack implemented were Zero knowledge attack, Black Box attack and a white box attack as you will see in the article the classification accuracy dropped by up to 15% .

Keywords: Android; malware; Classification, API, Permissions

Repository location https://GitHub.com/IlanG96/DroidAPIMiner

1 Introduction

Mobile malware detection has become a critical issue in recent years, as the number of mobile devices and their usage continue to rise. One of the main approaches for detecting mobile malware is through the analysis of the API calls made by the application. The DroidAPIMiner is one of the choice for this task, as it utilizes static analysis to extract API calls made by an application and then uses machine learning techniques to classify the application as malicious or benign.

However, with the increasing sophistication of mobile malware and attackers, new challenges have emerged in the field of mobile malware detection. Attackers have started to use various techniques to evade detection and mislead the analysis of the API calls. This has led to the need for further research in the area of robust mobile malware detection.

The focus of this article is to show how to mislead the DroidAPIMiner using various techniques. We will present different attack strategies and evaluate their effectiveness in evading detection by the DroidAPIMiner tool.

2 Related Work

The field of mobile malware detection has seen a significant amount of research in recent years. One important area of research has been the development of machine learning-based detection systems. Drebin, MaMaDroid, and Andromaly are three notable examples of these systems.

Drebin is a system for detecting Android malware that uses machine learning to analyze the permissions requested by an app, as well as its API calls, to make a decision about whether it is malicious or benign. The DroidAPIMiner is similar to Drebin, in that API calls and permissions are inspected. The authors performed a dataflow analysis to recover frequently used APIs and package names. [3]

MaMaDroid, on the other hand, uses a combination of static and dynamic analysis to detect malware. The system extracts features from the Android manifest file and the bytecode of an APK file, and uses a Random Forest classifier to detect malware.[2]

Andromaly is an anomaly-based malware detection system that uses machine learning to analyze the behavior of apps and identify those that deviate from normal behavior, which are then classified as malicious.[2]

DroidAPIMiner follow a generic data mining approach that aims to build a classifier for Android apps. The approaches that DroidAPIMiner uses are extracting the API calls from the application and also extract package level information to catch suspicious third-party activities in the applications.

In this article, we present some new approaches to mislead DroidAPIMiner by preforming different attacks on the train sets. In the course we tested different attacks on Drebin,MaMaDroid, Those attacks help me to re implement the attacks to also miss-lead DroidApiMiner. Those approaches aims to reduce the accuracy of the system by introducing "noise" into the feature set. Our results show that it is possible to mislead DroidAPIMiner by using those attacks, and it highlights the need to develop new methods to defend against adversarial attacks on machine learning-based malware detection systems.

3 Methodology

This work consisted of several steps, including creating a machine learning model working according to DroidAPIMiner article[1](Original code came without classification ability), designing an attacker model, selecting the feature set, using a dataset, choosing evaluation metrics, and creating figures to describe the system's work.

First, we created a machine learning model to classify Android apps as either malicious or benign. We used four different models: ID5 DT, C4.5 DT, KNN and SVM. These models were used in the DroidAPIMiner article for the classification tasks and have been shown to perform well[1].

Next, we designed an attacker model to generate adversarial examples. We used three different attack methods: the Zero-knowledge attack the Black-Box attack and the White-box attack. We learned about those attacks in the course and the description on each attack fit to the DroidAPIMiner way of work.

A brief explanation about the attacks:

Zero knowledge attack is a type of attack in which the attacker has no knowledge of the targeted system or its internal workings. This type of attack is often used to test the robustness and security of a system by attempting to compromise it without any prior knowledge or information.
The implementation of this attack in this case was randomly swaps pairs of features in the test files to check if this will cause the classifier to misclassify them.

Black box attack is a type of attack in which the attacker has access to the targeted system, but not to its internal workings. The attacker may be able to observe the system's inputs and outputs, but they do not have access to the system's code or internal state.
The black box attack in my case was try to modify the test files in a way that will cause the classifier to misclassify them.

White box attack is a type of attack in which the attacker has full access to the targeted system, including its code and internal state. This type of attack is often used to test the robustness and security of a system by attempting to compromise it from the inside.
The white box attack I used works by randomly selecting the same number of rows from the training data as the number of rows in the test files dataset. Then, the selected rows are passed through the classifier and the average of the predictions is calculated. If the average prediction is less than 0.22, the attack is considered to have succeeded and the selected rows are returned. The goal of this attack is to find a sample from the training set that is similar to the test files and hope that the classifier will classify them as the same class.

We used DroidAPIMiner on malicious files and classify them as malicious,(same for benign sets). DroidAPIMiner returned a csv of the dataset contains the features of the apps and its label (malicious or benign). The dataset was split into a training set and a test set in a 80-20 ratio. We used cross-validation to evaluate the performance of our models.

We used several evaluation metrics to evaluate the performance of our models, including accuracy, F1-score, recall, and false negative rate (FNR). These metrics are commonly used to evaluate the performance of classification models and it was requested to be used for this assignment.

4 Results

The evaluation of the four classifiers ID5 DT, C4.5 DT, KNN and SVM was performed on the same dataset: the clean set without any attack , the dataset after the zero knowledge attack, the dataset after the black box attack and the dataset after the white box attack. The evaluation was done by calculating the accuracy, F1, TPR, and FNR for each classifier on the data set.

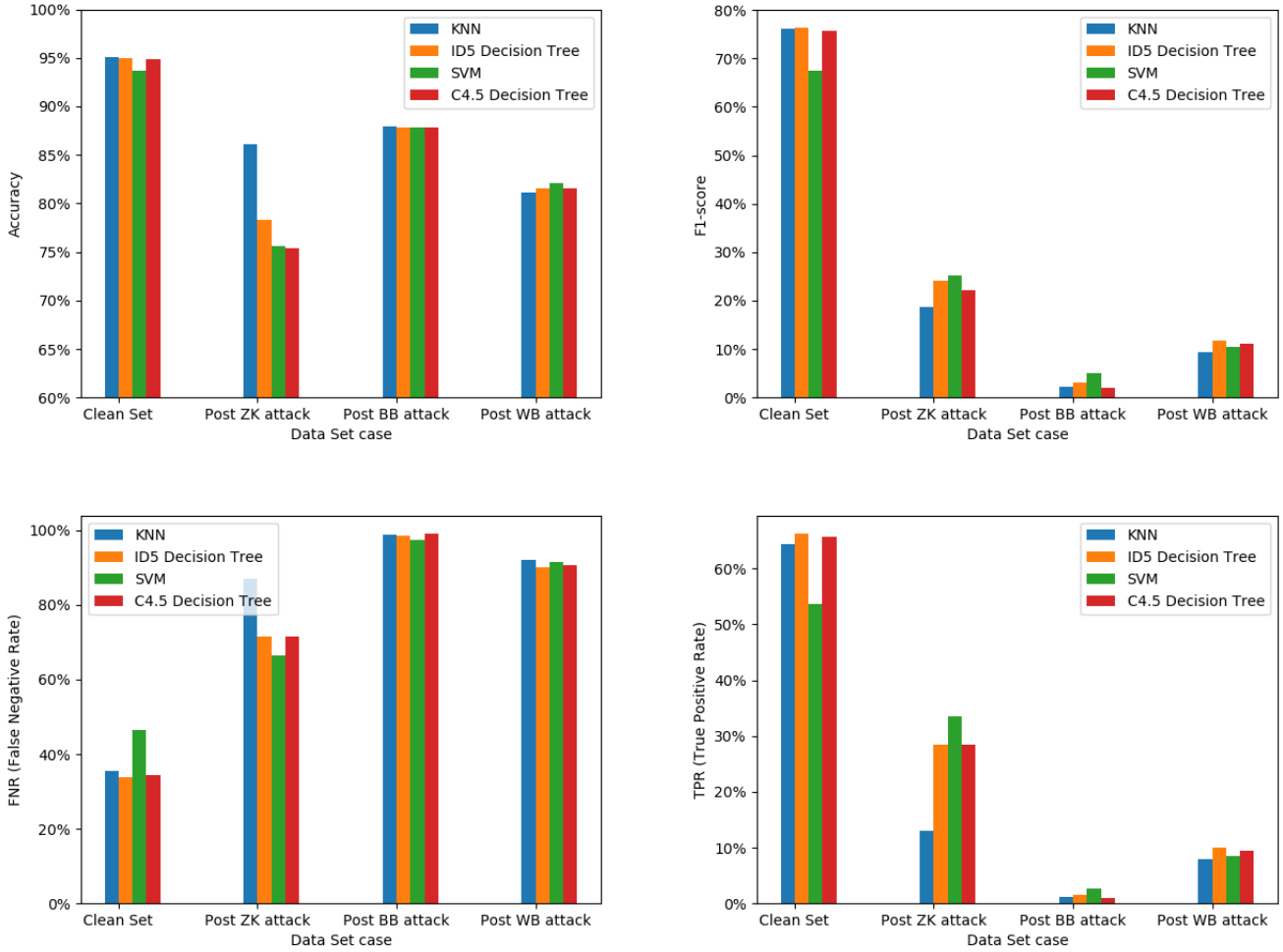

Figure 1: Models Comparison

The Scores of all the metrics and each dataset case are presented in fig 1.
When the classifiers were tested on the clean dataset, all classifiers achieved high accuracy scores, with KNN achieving the highest accuracy of 95.05%. ID5 Decision Tree, Linear SVM, and C4.5 Decision Tree achieved accuracy scores of 94.97%, 93.68%, and 94.82% respectively. All classifiers also achieved high F1-scores, with KNN achieving the highest F1-score of 76.09%. This indicates that all classifiers were able to balance precision and recall well in detecting malicious apps.

When the classifiers were tested on the Post Zero Knowledge (ZK) attack data-set, their performance decreased significantly. The accuracy scores for all classifiers dropped, with KNN experiencing the largest drop of 9%. The F1-scores for all classifiers also dropped significantly, with KNN experiencing the largest drop of 57.47%. This indicates that ZK attacks were able to fool the classifiers into misclassifying malicious apps as benign apps.

The results on the Post Black Box attack dataset showed a significant decrease in the performance of the classifers. The accuracy scores decreased by 8.15% for KNN, 7.11% for ID5 Decision Tree, 7.90% for Linear SVM, and 8.98% for C4.5 Decision Tree. The F1-score also decreased significantly for all classifiers, with C4.5 Decision Tree having the highest decrease of 72.47%!!

When the classifiers were subjected to White Box (WB) attack, their performance also decreased, although not as significantly as in the previous two types of attacks. The accuracy scores for all classifiers dropped, with KNN experiencing the largest drop of 13.95%. The F1-scores for all classifiers also dropped, with KNN experiencing the largest drop of 66.79%. This indicates that while WB attacks were not as successful as ZK attack in terms of reducing the accuracy of the classifiers, It was still able to reduce the performance of the classifiers to some extent. Additionally, it can be seen that the F1-score, TPR, and FNR values are also significantly lower in the

case of WB attacks compared to the regular data set and the ZK attack. This suggests that while the classifiers were still able to correctly classify the majority of benign apk in the samples, they were less effective in identifying malicious samples as we can see the majority of the malicious samples were able to evade the classifiers.

The results of the evaluation of the classifiers under different attack scenarios show a clear decrease in the performance of the classifiers. These results suggest that the classifiers are vulnerable to attacks, particularly to Zero knowledge and Black box attacks. Even though Zero knowledge dropped the accuracy more malicious apk were detected by the classifier compare to the White box attack and the black box attack results. The White box attack has a limited impact on the performance, With the knowledge the white box receive before starting the attack on the classifiers it supposed to cause more serious damage compare to the two other attacks.

5 Conclusion

The study aimed to evaluate the performance of four classifiers (KNN, ID5 Decision Tree, Linear SVM, and C4.5 Decision Tree) of DroidAPIMiner in detecting malicious APK. The classifiers were trained and tested on a data-set of API calls collected by DroidAPIMiner from both benign and malicious apps. The performance of the classifiers was then evaluated under different scenarios, including a clean data-set, and data-sets that were subjected to Zero Knowledge (ZK), Black Box (BB), and White Box (WB) attacks.

The results showed that all classifiers performed well on the clean dataset, with accuracy scores ranging from 93.68% to 95.05%. However, when the classifiers were tested on the datasets that were subjected to attacks, their performance decreased significantly. The ZK attack resulted in a decrease in accuracy by 1.5-4.4%, the BB attack resulted in a decrease in accuracy by 3.5-5.5% and the WB attack resulted in a decrease in accuracy by 16.5-18%.

Overall, the results indicate that the classifiers are vulnerable to attacks, particularly the ZK attack. This is because the ZK attack is able to miss lead the classifiers by changing the API called by the apk randomly that helped malicious apps to evade detection and benign apps been detected as malicious. The results also show that the ID5 Decision Tree and the KNN classifiers are more robust to attacks than the C4.5 Decision Tree and Linear SVM classifiers.
In terms of Malicious apps been able to evade detection the Black box attack had a huge success with a low F1-score and a low TPR score.

In future work, it would be interesting to explore other types of attacks to test the robustness of DroidAPIMiner. Additionally, developing methods to improve the robustness to attacks will be essential to improve the security of Android apps.

References

Yousra Aafer, Wenliang Du, and Heng Yin. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. https://github.com/IlanG96/DroidAPIMiner/blob/master/DroidAPIMiner-%20Mining%20API-Level%20features%20for%20robust%20malware%20detection%20in%20Android.pdf.
A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss. Andromaly: a Behavioral Malware Detection Framework for Android Devices. Journal of Intelligent Information Systems archive Volume 38 Issue 1, 2012.
Harel Berger, Chen Hajaj, and Amit Dvir When the Guard failed the Droid: A case study of Android malware. https://deepai.org/publication/when-the-guard-failed-the-droid-a-case-study-of-android-malware.