Kirin [1] 2.0: A New Era in Classifying Android Malware

from a 2009 classifier to a 2023 one, static classifier, ML, and much more

1st Dor Baram
student of Ariel University of samaria

2nd Omer Michael
student of Ariel University of samaria

***Abstract*—In this poster, we propose an improved version of the Android malware detection classifier, Kirin [1]. Our approach involves incorporating feature selection with random forest and using both random forest and XGboost to determine if an app is malicious or benign. Through this combination of techniques, we aim to improve the accuracy and efficiency of Android malware detection. Our evaluation results show that our approach outperforms the original Kirin classifier. The main insights of this poster are the importance of feature selection techniques, and the benefits of combining multiple machine learning algorithms for Android malware detection.**

I. Introduction

Android malware is a growing problem in the world of mobile technology. As the use of smartphones and tablets is increasing everyday, so too has the number of malicious apps designed to steal personal information, disrupt device functionality, spread malware to other devices and many many more. In response to this problem, researchers have developed various methods for detecting and analyzing Android malware.

One such method is the use of static analysis to identify malicious apps. This approach, which is employed by systems such as Kirin [1], involves analyzing an app's permissions and resources to extract features indicative of benign or malware application. While this approach has been shown to be effective in detecting malware, it is sometimes may not be able to detect new, previously unseen malware, as the Kirin classifier was written in 2009, it was a new way to detect malware. It used nine sets of permissions, which determined the classification of the application.

In this article, we will propose an improve version of the Kirin [1] classifier that addresses these limitations and bring Kirin to 2023. Our approach involves incorporating new feature selection with random forest, and determining if an app is malicious or benign using random forest and XGboost. Through this combination of techniques, we aim to improve the metrics of Android malware detection. In this paper, we will present our methodology and results of the evaluation that show how our approach outperforms the original Kirin [1].

II. Related Work

Drebin [2]: Drebin is a famous Android malware detection system that uses machine learning for its detection capabilities. The system employs various feature selection techniques to select the most relevant features for detection and uses machine learning algorithms, such as decision trees and SVM(support vector machine). The system has shown high accuracy and fast detection times in evaluations. However, there are also limitations to Drebin, such as its inability to detect new, previously unseen malware, and its reliance on a large dataset of labeled malware samples.

MaMaDroid [3]: MaMaDroid is an Android malware detection system that uses API call traces to identify malicious behavior. The system collects and analyzes API calls to extract features that are indicative of malware. These features are then used to train machine learning models that can detect malicious apps. The system has been shown to have high accuracy and efficiency in evaluations. However, one limitation of MaMaDroid [3] is that it can only detect malware that uses specific types of API calls, and it may not be able to detect new, previously unseen malware.

Andromaly [4]: Andromaly is an Android malware detection system that uses machine learning to identify anomalous behavior in Android devices. It extracts features from device logs, network traffic, and system calls to identify anomalies that may indicate malware. The system employs various machine learning techniques such as clustering, classification, and anomaly detection algorithms to detect malicious apps. It was shown to have a good performance in terms of accuracy and efficiency in evaluations. However, one limitation of Andromaly [4] is that it might generate false positives if the benign apps have similar patterns of anomalous behavior.

"DroidAnalytics" [5] is a framework for analyzing Android malware using static and dynamic analysis techniques. It combines the strengths of both techniques to provide comprehensive and efficient detection and analysis. The framework includes several modules for code, network, and behavioral analysis, and uses machine-learning techniques to improve detection. The evaluation showed a high detection rate and a low false positives rate. However, challenges faced include high computational costs and difficulty identifying malicious code. The paper suggests ways to overcome these challenges by using machine learning, and data mining techniques and incorporating other forms of machine learning such as deep

learning to improve accuracy and reduce reliance on large labeled datasets.

III. Methodology

Our methodology for improving Kirin [1] was to replace the outdated features that the classifier analyzed, that was combining a determined set of features and classifying with the combinations of them. the improved version that we got, was established by analyzing android malware involved in utilizing machine learning algorithms which is the contemporary way to look at the data. We used these algorithms to determine the best feature set through feature selection. Our data selection was limited to extracted permissions of the applications, the same as in the older version. This approach allowed us to work with a manageable amount of data, as using all of the 1000+ permissions would have resulted in a less precise outcome and greatly prolonged the processing time. We found that using only the top 110 most commonly used permissions yielded optimal results within a manageable amount of time.

ML Models: The Random-Forest-Classifier and XGB-Classifier are both popular machine-learning models used for classification tasks. The Random-Forest-Classifier is an ensemble learning method that utilizes multiple decision trees to make predictions. It combines the predictions of multiple trees, which helps to reduce overfitting and improve the overall measures of the model. The XGB-Classifier, on the other hand, is an implementation of the gradient-boosting algorithm using decision trees. The gradient boosting algorithm is known for its ability to handle large data sets and improve the performance of weak models, making it particularly useful for high-dimensional problems as in our case. Both models have been widely used in industry and have shown to be effective in a variety of applications. The overall way we used the algorithms is: we run the random forest on the features (permissions) and then take only the top 110 used ones, most of the permissions are used in a handful amount of applications, To ensure the robustness of our model, we implemented a cross-validation technique, running it over five random slices of the data. Then we proceed to run the XG-boost algorithm five times on the data, and that is how we teach the classifier the way to distinguish between benign applications and malicious ones. ML algorithms that we tried and got worse time/results are: Decision-Tree-Classifier, Ada-Boost-Classifier, Extra-Trees-Classifier, and Recursive-Feature-Elimination.

Datasets: The applications we used to run the classifier on are from Crystal ball [6].

The ratio we used with benign applications and malicious applications is 90-10 and it is suggested by 'TESSERACT' [7].

Performance metrics, such as True Positive Rate (TPR), False Negative Rate (FNR), F1 Score, and Accuracy, are utilized to evaluate the precision of a machine learning model. TPR is determined by the proportion of true positives correctly identified by the model to the total number of positive examples in the data set. Conversely, FNR is determined by the proportion of false negatives incorrectly predicted by the model to the total number of negative examples in the data set. The F1 Score is a composite metric that combines both precision and recall and is used for evaluation purposes. It takes into account both TPR and FNR when calculating its value. Lastly, Accuracy measures the ability of a classifier to correctly predict classes and provides an overall assessment of the reliability and precision of a given machine learning algorithm on unseen data points.

IV. Results

The results of our analysis show that our improved classifier is able to effectively detect android malware with high accuracy and measurements. Table 1 shows the metrics for the classifier, including accuracy, F1 score, and TPR. The results demonstrate that our improved classifier outperforms the old classifier, Kirin, in all metrics.

Measurements	Original Kirin [1]	New Kirin	Change
TPR	0.478	0.659	+ 0.181
FNR	0.522	0.341	- 0.181
F1	0.474	0.782	+ 0.308
Accuracy	0.870	0.930	+ 0.060

TABLE I
TABLE OF TPR (TRUE POSITIVE RATE), FNR (FALSE NEGATIVE RATE), F1 SCORE, ACCURACY FOR ORIGINAL [1] AND NEW KIRIN

In addition, we also show the results of our classifier in the form of a ROC curve in Figure 1. The ROC curve illustrates the trade-off between the true positive rate (TPR) and false positive rate (FPR) of the classifier. The curve for our improved classifier is closer to the top-left corner, which indicates that it has a higher TPR and lower FPR compared to the old classifier, which is under the 0.5 line.

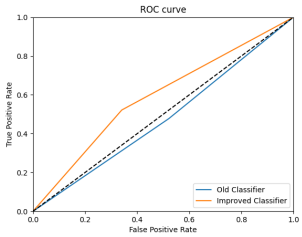

Fig. 1. ROC curve for the classifiers

The results of our analysis indicate that the implementation of feature selection via random forest and the use of XG Boost algorithm were key in improving the performance of the classifier. The improved classifier was able to effectively detect android malware with high accuracy and measurements.

V. Conclusion

In this paper, we presented our work on improving the accuracy and measurements of the original classifier Kirin, for android malware analysis. We implemented feature selection via random forest, which effectively reduced the number of features from above 1000 to about 100. The features we used were the permissions requested by the applications, which were found to be highly indicative of whether the application is benign or malicious. Additionally, we ran XG-Boost algorithm five times over the data to further improve the performance of the classifier, and to mitigate over fitting via cross validation. The classifier was then able to effectively determine between benign and malicious apps with high accuracy, as well as high F1 and TPR measurements. We ran the classifier on over 60k tagged applications, and it took under an hour to complete. The results of our analysis show that our improved classifier is able to effectively detect android malware with improved accuracy and measurements compared to the old classifier. In conclusion, our work demonstrates the effectiveness of using features selection from as application permissions and run advanced machine learning algorithms to improve the accuracy and other measurement of android malware classifiers, even when applied to large sets of data.

References

[1] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245, 2009.

[2] 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.

[3] 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.

[4] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[5] Min Zheng, Mingshen Sun, and John CS Lui. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 163–171. IEEE, 2013.

[6] Harel Berger, Chen Hajaj, Enrico Mariconti, and Amit Dvir. Crystal ball: From innovative attacks to attack effectiveness classifier. *IEEE Access*, 10:1317–1333, 2022.

[7] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 729–746, 2019.