Lightweight Malware Detection based on Machine Learning Algorithms and the Android Manifest File

Monica Kumaran
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, U.S.A.
monica.kumaran@berkeley.edu

Wenjia Li
School of Engineering and Computing Sciences
New York Institute of Technology
New York, U.S.A.
wli20@nyit.edu

Abstract — This study aims to learn if the Android manifest file provides enough information to classify an app as malicious or benign. In particular it compares the efficacy of using requested permissions versus inter-app intent communication. It also improves static malware detection by comparing and refining different machine learning algorithms on the manifest file dataset. I find that a Cubic Support Vector Machine (SVM) algorithm is the most accurate classifier for the complete dataset with a 91.7% accuracy. I also find that combining intent filters with requested permissions improves the classifier, but intent filters are not enough to base a classifier on while permissions are.

Index Terms—Malware detection, Android, static analysis, machine learning, manifest file.

I. Introduction

Malware greatly harms victims by collecting and selling personal data to advertisers and phishers, selling mobile device identification numbers, or initiating expensive premium calls and texts without user permission. As the most popular mobile operating system, the open-source Android platform is particularly susceptible to malicious apps. In fact ninety-eight percent of all mobile malware is targeted at the Android marketplace [1]. To understand the massive influence of the research area, as of 2016 the Android marketplace hosts 2.2 million apps and accounts for 65 billion total app downloads [2]. The malicious sliver of the market has the potential to affect millions of users. Current malware detection methods normally use one of three approaches: static analysis, dynamic analysis, and machine learning algorithms. This study combines static analysis and machine learning algorithms.

To detect Android malware most static approaches use requested permissions as the backbone of analysis. This study aims to learn if parsing only the Android manifest file is enough information to classify an app as malicious or benign. Another purpose of the study is to discern if permission sets or app intent filters are more effective indicators of malware. Finally, the study compares several different machine learning algorithms to find one which creates the most effective classifier.

II. Related Work

The DREBIN program detects 94% of malware with a false-positive rate of 1% and takes around 10 seconds to run on a smartphone [3]. It is purely a static analysis but combines data from both the Android manifest file and embedded API calls. It creates a matrix of indicator variables for each possible feature including hardware used, requested permissions, app component names, and intent filters which control information shared between components. Data from API calls includes restricted API calls, used permissions, suspicious API calls, and network addresses. The researchers used a linear support vector machine (SVM) with a ten fold cross-validation to separate benign from malicious apps, while storing weights to track which features are suspicious. Limitations are that malware examples are not as common as benign examples, and hackers can obfuscate malware code to fool the detection system.

Another related approach is Xu et al.'s detection based on inter-component communication [4]. Malicious apps can generate an explicit intent and manipulate other apps into installing malware. Explicit intents are normally used within one app and specify components, while implicit intents ask for general actions. The researchers extracted intents filters and objects from each app and used a SVM with a ten-fold cross validation to detect malware. The program was run over 5,264 malware and 12,026 benign apps. The malware detector achieved an accuracy rate of 97.4%, outperforming the benchmark by 10% and having a lower false positive rate. It was also discovered that malware tends to have more broadcast filtered intents and dynamic intent filters, for example battery change, boot complete, package add, are often exploited. Limitations are that the ICCDetector does not dynamically inspect malware and can be bypassed with mimicry & pollution attacks. Both studies can be improved upon with experimentation across different types of machine learning algorithms.

III. Methodology

Android apps are stored as .apk files, which were decompiled using the open source apktool. For each app the manifest file, a .xml file, was further decomposed into permissions and intent filters using Python's ElementTree XML API. Information was stored in a matrix of indicator variables with each row representing an Android app and

columns representing what features it declares. The 183 features extracted were from three categories:

A. Requested Permissions. Android apps must request user consent to access functionalities such as location, camera, or contacts. They must also list all their requests in the manifest file. A complete list of Android permissions was acquired from the Android developer website. This accounted for 154 features.

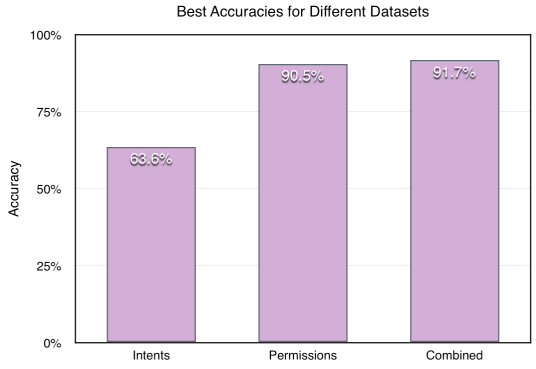
B. Declared Permissions. Android apps can create their own permissions, which can be used as an extra security layer against other apps accessing their data. A single variable was marked true if an app created any of its own permissions.

C. Intent Filters. Intents are used to request actions from other app components. Apps can send implicit intents to other apps requesting information. Intent filters tell the system which implicit intents the app can handle and what overall phone states the app wants to know. A list of common intent filters was acquired from the Android developer website. This accounted for 28 features.

After extracting information from the original dataset, different machine learning algorithms were trained on a training set of 500 benign and 500 malicious apps. A ten fold validation scheme was used to determine accuracy.

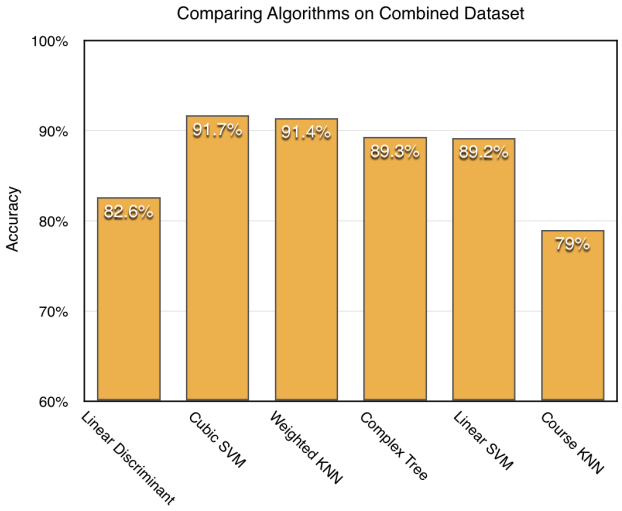
IV. Discussion

Machine learning algorithms were trained on three different datasets to discern if requested permissions or intent filters are more effective indicators of malware. One was of only intent filters, one was only requested and declared permissions, and one was both feature sets combined.



As Figure 1 illustrates, at 63.6% accuracy the intent filter set does not provide enough information to classify apps. However, an accuracy over 50% is still a strong enough result to warrant further exploration. When combined with permissions it increases the permissions' classifier accuracy by 1.2%. It is interesting to note that each best accuracy was achieved using a different machine learning algorithm. The results for the intent filters, permissions, and combined were achieved using a linear subspace discriminant, weighted k-nearest neighbors, and cubic support vector machine algorithm, respectively. Further exploration with differently sized datasets may yield different results.

The next goal of the study is to compare different machine learning algorithms over the complete, robust dataset.



As Figure 2 details, the cubic support vector machine (SVM) generates the best accuracy of 91.7%. However, that accuracy is not too different from the next two best algorithms, which had accuracies of 91.4% and 89.3% respectively.

V. Conclusion

It is possible to use only the manifest file to create a viable classifier. Using intent filters as well as permissions is a valid area of exploration based on the 1.2% increase of classifier accuracy. It is especially important considering that current malicious apps often use implicit intents to manipulate other apps and bypass traditional malware detection. Further refinement to the algorithms and dataset is necessary.

VI. Limitations

The classifier only depends on the manifest file. Even if the classifier is effective now, the malware landscape is constantly shifting and may render the approach obsolete in a few years. In general static approaches can be fooled by code obfuscation.

Acknowledgment

I would like to thank Dr. Wenjia Li, Kendra Campbell, the REU program at the New York Institute of Technology, and National Science Foundation Grant No. CNS-1559652 for supporting me in this endeavor.

References

[1] D. Neal, "Android is target for 98 percent of all mobile malware | TheINQUIRER", http://www.theinquirer.net, 2014. [Online]. Available:

http://www.theinquirer.net/inquirer/news/2331127/android-is-target-for-98-percent-of-all-mobile-malware. [Accessed: 08- Aug- 2016].

[2] "Topic: Android", www.statista.com, 2016. [Online]. Available: http://www.statista.com/topics/876/android/. [Accessed: 08- Aug- 2016].

[3] D. Arp, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket", 2014.K. Elissa, “Title of paper if known,” unpublished.

[4] K. Xu, Y. Li and R. Deng, "ICCDetector: ICC-Based Malware Detection on Android", IEEE Trans.Inform.Forensic Secur., vol. 11, no. 6, pp. 1252-1264, 2016.