Android Malware Detection – Project Proposal

Matthew Chen, Mark Faynboym, Jasper Tsai

1 Introduction

Android holds 74% of global mobile OS user share and has 2 billion active devices (Mathur et al., 2021). Further, its open-source nature makes it especially prone to malicious software as 97% of all mobile malware is targeted at Android (Akbar et al., 2022). In this project, we build upon the work of Mathur et al., 2021 by using data analysis and supervised machine learning to classify malicious software using native and custom permissions as the input features. The questions that we address in this project include:

1. How can we best detect malicious software using readily available features (i.e. application permissions), building upon the state-of-the-art research?
2. What permissions (or set or permissions) are the most frequently occurring in malicious software?
3. Can we improve established performance using complex statistical models such as deep neural network or ensemble stacking?

2 Data

The data we are using is sourced from the NATICUSdroid Project (Mathur, 2022), which includes data from 29,000 benign and malware applications collected between 2010 and 2019. The benign applications in this dataset include 14,630 observations from Androzoo which have been rated as benign by VirusTotal. Data for malware applications were sourced from Argus Lab's Android Malware Database with 14,700 applications randomly selected out of the 24,500 available so that the overall dataset is balanced. In total, the dataset includes 86 native and custom Android permissions, represented as 0-1 dummy variables. Our target variable is whether the application is malware and is also represented as a 0-1 dummy variable.

3 Literature Review

Akbar et. al, (2022) took a similar approach to malware detection and used supervised machine learning, specifically Support Vector Machine (SVM), Random Forest (RF), and Naïve Bayes (NB), on 10,000 manually collected benign and malware applications. The authors sourced 5000 instances of benign software from the Google Play Store and 5000 instances of malware software from VirusShare. This assumes that all benign applications are similar to those in the Google Play Store, and that all applications in the Google Play Store are benign (there have indeed been reported instances of malware appearing on the platform). Similarly, the authors assume that VirusShare is representative of malware applications as a whole. For feature selection of permissions, the authors balanced domain knowledge of including dangerous permissions as well as selection based on feature importance scores from random forest. Notably, SVM and random forest achieved accuracies of 89.7% and 89.96%, respectively, each averaged over 10 folds in cross validation. The F1-scores were similarly high.

Another similar reference is Mathur et. al, (2021), which also performed malware detection using the NATICUSdroid dataset described above. We again make the assumption that the sourced benign and malware applications are representative of benign and malware applications as a

whole. Further, instead of assuming that all Google Play Applications are benign, the NATICUSdroid dataset was validated by benign ratings from VirusTotal. The authors here experimented with 8 different supervised classification algorithms, with Random Forest (97% accuracy) and k-Nearest Neighbors (96% accuracy) among the highest performing. Again, the F1-scores were similarly high.

4 Methodology

4.1 Feature Selection and Dimensionality Reduction

Since there are 86 different permissions in the dataset, it is unlikely that all of them are important for the classification task. In selecting features, the goal is to balance domain knowledge, statistical selection, and interpretability.

For example, Android permissions are categorized as normal, dangerous, signature, and signature privileged (Mathur et al., 2021). Normal permissions are considered benign and are granted by default, while dangerous permissions are sensitive and many require user approval. In selecting variables, we want to consider the sensitivity of the permissions omitted.

In our data driven approach, we will use feature importance from a Random Forest classifier, which ranks a variable's contribution to decreasing the impurity score across the forest so that more important variables for the classification task will be ranked higher (Hastie et al., 2008). Another advantage of using random forest for feature selection is that it preserves the interpretability of the features. Overall, our strategy to perform feature selection will be to select the top n import features from the result while also taking the sensitivity of omitted permissions into account, as discussed above. If any high multicollinearity exists in the selected subset of features, we will further choose feature pairs manually. It is important to consider high multicollinearity in the model since it may lead to high variance (overfitting) or unstable estimators.

Another dimensionality reduction strategy worth considering is Principal Components Analysis (PCA), which transforms the data into uncorrelated features that are ordered in the variance they explain in the data. Unlike the former strategy, PCA does not preserve the interpretability of the original input features. However, if the data has a strong correlation structure and the eigenvalues of the covariance matrix decay quickly, PCA could prove to be an effective data compression technique upstream of further predictive modeling.

4.2 Predictive modeling

Previous research has shown that random forests and other supervised algorithms are successful at detecting malware using application permissions. We note that undetected malware (false negative/Type 2 error) is destructive since it leaves the user vulnerable, while on the other hand false positives (Type 1 error) is irritating for the user. While Type 2 error may be more severe in this context, we believe that it is important to explore ways to further enhance predictive ability overall. Thus, we will first validate the results from Mathur (2021) using their most successful algorithms, namely, random forest and k-nearest neighbors. We will also train a logistic regression model as an additional baseline. Then, as an attempt to improve established performance, we will experiment with using deep neural networks and ensemble stacking.

In model stacking, an algorithm is used to combine the strengths of several other high performing models into a single improved model. Often times, the final ensemble model is higher performing than any single model incorporated. In fact, it can be shown that the stacked model (represented by a weighted output of ensemble member predictions) on square error loss will never be worse than a single model member in expectation at the population level (Hastie et al., 2008). This method is appropriate to try since previous research has already produced a variety of high performing models as potential ensemble members.

Finally, we will also train a feed-forward neural network on this classification task to compare its results. Neural networks are complex statistical models that have shown to outperform many traditional methods on a variety of tasks. This is due to the fact that neural networks have an arbitrarily high degree of freedom, given that there is enough data to train and take advantage of them, i.e. the Universal Approximation Theorem for neural networks state that there exists a neural network such that any well behaved function can be approximated with arbitrarily low error (Goodfellow et al., 2016). This method is appropriate for this project since the data size is reasonably large (~30,000 observations) and has not yet been applied in the literature on NATICUS.

5 Expected Results

Given the past success of permissions-based malware detection using supervised learning, we would expect that our models also perform at least as well (Akbar et al., 2022; Mathur et al., 2021). However, given the severity of undetected malware, our project explores the possibility of further increasing predictive performance using deep learning and ensemble stacking.

6 References

Akbar, F., Hussain, M., Mumtaz, R., Riaz, Q., Wahab, A. W. A., & Jung, K. H. (2022). Permissions-Based Detection of Android Malware Using Machine Learning. *Symmetry*, *14*(4). https://doi.org/10.3390/sym14040718

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*.

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*.

Mathur, A. (2022). NATICUSdroid (Android Permissions) Dataset. In *UCI Machine Learning Repository*. https://doi.org/https://doi.org/10.24432/C5FS64

Mathur, A., Podila, L. M., Kulkarni, K., Niyaz, Q., & Javaid, A. Y. (2021). NATICUSdroid: A malware detection framework for Android using native and custom permissions. *Journal of Information Security and Applications*, *58*. https://doi.org/10.1016/j.jisa.2020.102696