Android Malware Detection – Project Proposal

Matthew Chen, Mark Faynboym, Jasper Tsai

# Introduction

Android holds 74% of the world’s 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. In this project, we build upon the work of Mathur et al., 2021, in using data analysis and machine learning to classify malicious software on Android based upon native and custom permissions as the features. The questions that we are trying to 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?

# 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. On the other hand, 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 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 or not the application is malware, and is also represented as a 0-1 dummy variable.

*Summarize the results and the methods of at least two existing papers that addressed problems similar to the one you are trying to address. For each paper, clearly state the authors’ implicit or explicit assumptions.*

# Methodology

## 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 more 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. In other dimensionality reduction techniques, such as principal components analysis, the dimensionally reduced features no longer retain their original interpretation. 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.

## Predictive modeling

Previous research has shown that random forest is highly skilled at classifying malware on the NATICUSdroid dataset with f-score of 0.96 and an accuracy of 97% (Mathur et al., 2021). However, undetected malware is extremely malicious and destructive, while on the other hand false positives are irritating for the user. For this reason, it is important to explore ways to further enhance model performance.

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, we will explore using neural networks and ensemble stacking in an attempt to improve established performance.

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 a similar problem.

*Based on your reading of existing literature, what results would you anticipate for your project? Justify your prediction, referencing existing results.*