Detecting Malicious Android Apps Using App Permissions

Tim Salomonsson V00807959
Matthew Cairns V00709952

SENG 474 Final Project
April 5th, 2019

Introduction

The goal of this project is to predict whether an Android application is malicious, given the app's required permissions. The main repository for Android applications is the Google Play Store. Google "enforces stringent checks for new applications, but updates are made over time, […] and the level of checking may be reduced" [1] Given this, it is often simple for malware to enter the Play Store [2]. Therefore, an efficient algorithm is needed to flag possible malware for further analysis. This project implements a malware prediction algorithm trained using a human curated dataset of malware and goodware. Prediction relies on the hypothesis that some malicious applications commonly use similar sets of permissions. Regression analysis is used to identify possible malicious apps based on their permission sets.

Dataset

The project dataset was provided to by Koodous, a Android malware research platform [3]. The dataset consists of 1.5TB (~110k) Android APK (Android Application Package) files; 60% are identified as malware and 40% goodware. The permissions required by an application are extracted from the application manifest XML file. The features of each app are binary variables, wherein the variable is 1 if the application requires that permission, otherwise 0.

The dataset requires a few filters to sanitize the input. First, all improper APKs are removed. Then the set of all permissions across all applications must be sanitized. All non-English and non-ASCII encoded strings are removed. Then malformed permissions outside the pattern "<package>.<NAME>" are removed. Finally, trailing whitespace is removed and all permissions converted to lowercase. The final dataset consists of 6000 apps each with 3359 possible permissions. We used this final dataset to train our model for logistic regression.

Machine Learning Model

Stochastic gradient descent for linear regression was initially used to create the model parameters as we were most familiar with this machine learning algorithm. But, since the dependent variable, i.e malware or not, is a binary variable it was soon apparent that this

approach was insufficient. Instead, logistic regression was used to return a probability prediction between 0 and 1 for each prediction.

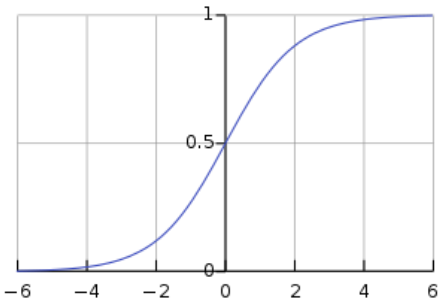

Figure 1: The logistic function[34].

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$

Figure 2: The sigmoid function[34]

The logistic function used is the sigmoid function, a special case of the logistic function. The domain of the sigmoid function is all real numbers, and returns a value from (0, 1), see figure 1.

Training

Stochastic gradient descent was chosen as the learning algorithm because it is less computationally intensive, given the large size of the dataset. The algorithm uses an adaptive learning rate function known as a bold driver [4]. This function increases the learning rate by 5% if loss decreases, otherwise the learning rate is decreased by 50%. The bold driver allows for a higher initial learning rate and reaches minimal loss in fewer iterations than with a static learning rate - choosing the number of epochs becomes more iterations in this case as we found the model to overfit after fewer epochs using this method.

Validation

The samples were split into equal sized training and validation sets. The model parameters are estimated with the training set. Then the model is evaluated through predictions on the validation set. Furthermore, a small dataset from Kaggle.com [5] is used to further test the model.

Results

Three performance metrics are used to evaluate the effectiveness of the machine learning model: accuracy, precision, and recall (see Table 1).

Algorithm	Dataset	# samples	# features	Accuracy	Precision	Recall
Linear regression	Koodous	3000	3359	0.748	0.863	0.578
	Kaggle	300	220	0.633	0.990	0.782
Logistic regression	As above			0.941	0.935	0.500
				0.852	0.854	0.501

Table 1: Model results.

Linear regression has 75% accuracy, compared to 94% with logistic regression. This means logistic regression correctly predicted 94% of applications. Also, logistic regression has a high precision; 93% of identified malicious apps are actually malicious. However, a low recall means only 50% of malicious apps are identified by logistic regression. The logistic regression model accuracy and precision fell slightly when tested with the dataset from Kaggle.com. The linear regression model lost accuracy but gained precision and recall.

Conclusion

In conclusion, our goal was to utilize a large dataset of Android applications flagged as malicious or not malicious to train a model to detect new malicious applications. We were able to verify that permissions have some relationship to app maliciousness by achieving a high detection accuracy.

Accuracy may be improved in future iterations through better data cleanup. Such as filtering permissions which are known to have no significant impact on maliciousness, and filtering out extremely rare permissions occurrences. More data could also be utilized, such as the apps description, author, country of origin and others.

References

[1] D. Palmer, "This data-stealing Android malware infiltrated the Google Play Store, infecting users in 196 countries," ZDNet, 04 January 2019. [Online]. Available: https://www.zdnet.com/article/this-data-stealing-android-malware-infiltrated-the-google-play-store-infecting-users-in-196-countries/. [Accessed 29 March 2019].

[2] Check Point Research, "Malicious Flashlight Apps on Google Play," Check Point Research, 05 January 2018. [Online]. Available: https://research.checkpoint.com/malicious-flashlight-apps-google-play/. [Accessed 29 March 2019].

[3] n.a, "Koodous," Koodous, 2019. [Online]. Available: https://koodous.com/. [Accessed 29 March 2019].

[4] n.a, "Momentum and Learning Rate Adaptation," n.d. [Online]. Available: http://www.willamette.edu/~gorr/classes/cs449/momrate.html. [Accessed 29 Match 2019].

[5] C. Urcuqui and A. Navarro, "Machine Learning Classifies for Android Maleware Analysis," *Communications and Computing (COLCOM)*, pp. 1-6, 2016.