Machine Learning based Android Malware Detection using Server

Rinoy R K

Department of Computer Science and Engineering

Anna University Chennai 600025, India

rinoyr@gmail.com

***Abstract*— Android occupies a major share in the mobile application market. The alarming growth rate of malicious apps has become a serious issue that sets back the prosperous mobile ecosystem. Android mobiles have become an easy target for the attackers. The main reason is the user ignorance in the process of installing and usage of the apps. Classification based on machine learning is better than any algorithm to detect new malware apps. In this project machine learning algorithm is used for detection of android malware. Along with the selected permissions different features are extracted from apps which are used for training the model to improve accuracy and to build a powerful model to predict whether the app is good or malicious for new apps. Selected features are given a score based on its use in benign and malware apps and model is generated using the SVM machine learning algorithm. It is implemented in server/cloud with API so that any authorized user can send an APK file and get the result with high accuracy.**

***Keywords*— Android Malware Detection, Machine Learning, Android permissions,**

I. INTRODUCTION

Android is currently the most used smart-mobile device platform in the world [1]. As of now, there are nearly 2.5 million apps available for downloading from Google Play, and more than 65 billion downloads to date [2]. Android platform is growing rapidly due to its performance and support for third party apps. Android allows users to install applications from unverified sources such as torrent, direct downloads, third party stores without proper verification from the source or downloaded device. Unfortunately, the popularity of Android also creates interests from cyber-criminals who create malicious apps that can steal sensitive information and compromise systems. . In 2016 alone, over 3.25 million new malicious apps have been uncovered. This roughly translates to an introduction of a new malicious Android app every 10 seconds [3]. These malicious apps are created to perform different types of attacks in the form of trojans, worms, exploits, and viruses. Each type of malicious apps has at least 50 variants, which makes it extremely challenging to detect them all.

Malware have been used as a means for conducting cyber-attacks for decades. Wide adoption of smartphones, which store lots of private and confidential information, made them an important target for malware developers. Android as the dominant mobile operating system has always been an interesting platform for malware developers and lots of Android malware species are infecting vulnerable users every day which make manual malware investigation an impossible mission. To address these elevating security concerns, machine learning algorithm can be used.

Machine learning is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to learn and progressively improve performance on a specific task from data, without being explicitly programmed. In this project machine learning is used to detect goodware and malware Android applications by learning patterns from the features of goodware and malware of training apps.

II. MALWARE DETECTION

Any software or a program with bad, unauthorized, and illegal behaviour can be called as malware. A mobile user can hardly identify the intensions of the application based on its permissions. The malware apps typically are programmed by the attackers to steal credentials of financial transactions, emails, social networking information, key stroke information, camera images, local file system information etc [4][5]. Therefore a mobile security application is needed that can detect and identify genuine and malignant apps based on permissions that the apps request.

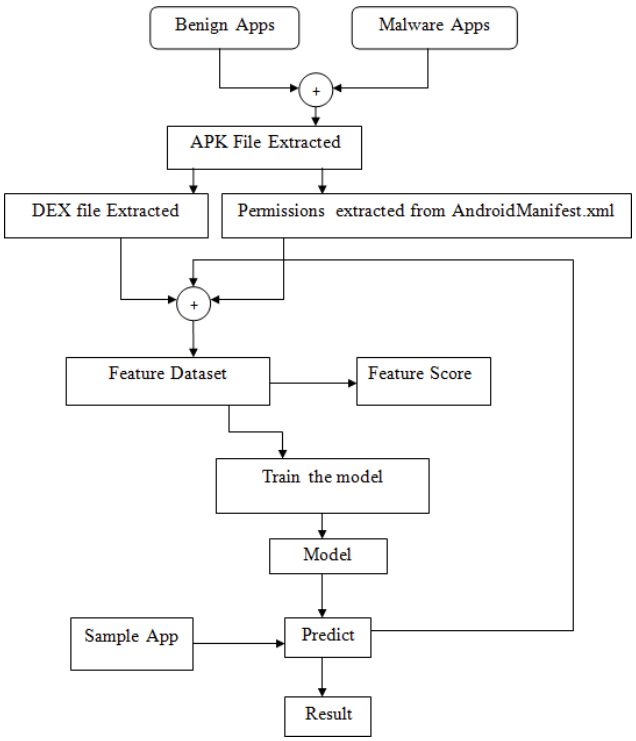

Fig. 1 Block diagram of proposed malware detection system.

A. APK File Extraction

Android Package is the package file format used by the Android operating system for distribution and installation of mobile apps and middleware. APK files are a type of archive file, specifically in zip format-type packages, based on the JAR file format, with .apk as the filename extension. APK files can be installed on Android-powered devices just like installing software on a PC. When a user downloads and installs an Android application, from either an official source such as the Google Play Store, or from an unofficial site, they are installing an APK file on to their device. A user or developer can also install an APK file directly to a device that is, not via download from the network from a desktop computer, using a communication program such as adb, or from within a file manager app in a process known as side loading.

APK file is a package file which can be extracted, contains all of the program's code such as .dex files, resources, assets, lib, certificates and Androidmanifest.xml file. Dex files are the classes compiled in the dex file format understandable by Dalvik Virtual Machine.

In order to use machine learning algorithms in classification of android malware, first decompress the application package to extract the dex file and then the permissions of the app are to be extracted from Androidmanifest.XML.

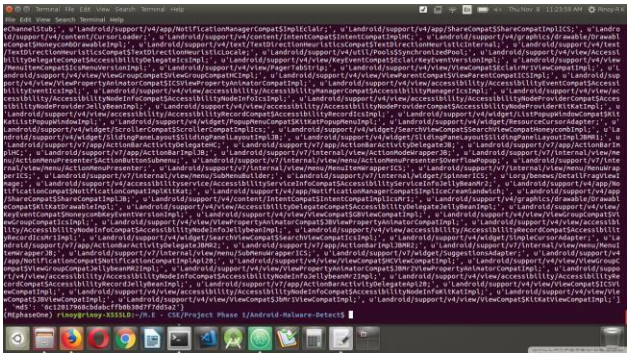

Fig. 2 Extracted Data

B. Feature Selection and Model Training

Extracted features are selected using the feature weight, the coefficient value from Linear SVC. The objective of a Linear SVC is to fit to the data you provide, returning a –best fit‖ hyperplane that divides, or categorizes, the input data. After getting the hyperplane, the model can be trained so that new input data can feed some features to the classifier to get the predicted output [10].

SVC and NuSVC are similar methods, but accept slightly different sets of parameters. On the other hand, LinearSVC is another implementation of Support Vector Classification for the case of a linear kernel [8].

Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples [9].

A database is created by placing a 1 if the permission is present in the Androidmanifest.XML file and a 0 if not.

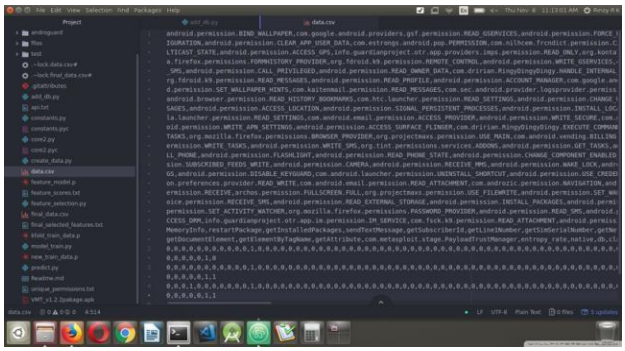

Fig. 3 Selected Features

Following features are extracted from APK file:

- Android version
- Version name
- Max SDK
- Min SDK
- Libraries
- Filename
- Target SDK
- MD5
- SHA256
- Permissions
- Activities
- Providers
- Services
- Class names
- Method names
- Field names
- API calls

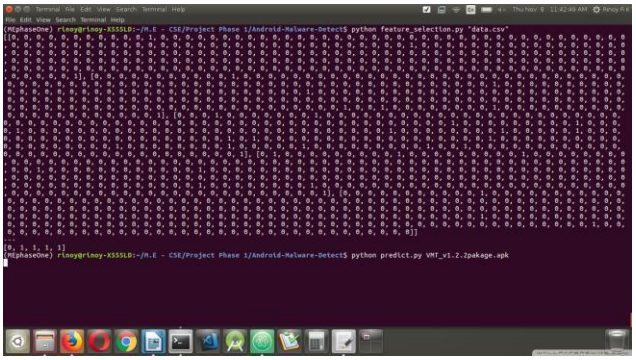

Fig. 4 The extracted features are assigned 0 or 1

C. Prediction

The model generated is used to predict whether it is malware or not for new input APK files. It uses Linear SVM machine learning to classify each APK files. Model is updated for every new files send to prediction.

III. SERVER IMPLEMENTATION

The Android malware detection system is implemented in server or cloud environment so that an authenticated user can use it as a service via an API [6]. An authenticated user can send an Android application to the server/cloud to detect whether the app is malware or goodware. The server already has the model trained using training apps or unknown apps classified by the system.

Server implementation would improve accuracy as well as performance eventually, because the model is learned from a variety of Android apps received from various users. For every Android app send to the server is extracted, classified and result is returned then learning model is updated with the new data.

IV. CONCLUSIONS

This paper has attempted to improve the accuracy in malware app detection using selected features along with most significant permissions, selected features and SVM algorithm. It can also improve the learning model since the learning is all done in the server where every authenticated user can send and get result about their APK file. Every new prediction will improve the model prediction accuracy. Thus this method can improve security and can be used to prevent the malware apps from installing into the Android.

To improve performance and accuracy deep learning techniques can be used also extracting more features from the APK files can improve accuracy of the learning model [7].

REFERENCES

[1] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, ―Appcontext: Differentiating malicious and benign mobile app behaviors using context,‖ in Software engineering (ICSE), 2015 IEEE/ACM 37th IEEE international conference on, vol. 1. IEEE, 2015, pp. 303–313.

[2] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, ―Textdroid: Semantics-based detection of mobile malware using network flows.‖

[3] Z. Li, L. Sun, Q. Yan, W. Srisa-an, and Z. Chen, ―Droidclassifier: Efficient adaptive mining of application-layer header for classifying android malware,‖ in International Conference on Security and Privacy in Communication Systems. Springer, 2016, pp. 597–616.

[4] J. Z. L. H. P. S. Y. Lichao Sun, Xiaokai Wei and W. Srisa-an, –Contaminant removal for android malware detection systems,‖ in Proceedings of IEEE International Conference on Big Data, 2017.

[5] L. Sun, Y. Wang, B. Cao, P. S. Yu, W. Srisa-an, and A. D. Leow, –Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning,‖ in Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2017.

[6] J. Li, Y. Zhang, X. Chen, and Y. Xiang, –Secure attribute-based data sharing for resource-limited users in cloud computing,‖ Computer and Security, vol. 72, pp. 1–12, 2018.

[7] P. Li, J. Li, Z. Huang, T. Li, C. Gao, X. Liu, and K. Chen, –Multi-key privacy-preserving deep learning in cloud computing,‖ Future Generation Computer Systems, vol. 74, pp. 76–85, 2017.

[8] Fan, Rong-En; Chang, Kai-Wei; Hsieh, Cho-Jui; Wang, Xiang-Rui; Lin, Chih-Jen (2008). "LIBLINEAR: A library for large linear classification" (PDF). Journal of Machine Learning Research

[9] Hsu, Chih-Wei & Lin, Chih-Jen (2002). "A Comparison of Methods for Multiclass Support Vector Machines" (PDF). IEEE Transactions on Neural Networks.

[10] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018