A New Decision Tree Ensembles Method for Fake Apps Detection in Android Platform

Huan Ren[1, 2, a], Wei Zhang[1, b], Qingshan Jiang[1, c]

[1] Shenzhen Institutes of Advanced Technology,

Chinese Academy of Sciences, Shenzhen 518055, China

[2] School of Software Engineering,

University of Science and Technology of China, Hefei 230051, China

[a]email: ivancanf@mail.ustc.edu.cn, [b]email:wei.zhang1@siat.ac.cn, [c] email: qs.jiang@siat.ac.cn

Keywords: Smartphone Security; Android fake Apps; Feature-based; Ensemble Learning

Abstract. The sharp increase in the number of smartphones makes the fake Apps in Android platform to be an urgent issue. Many approaches have been proposed to defend against these fake Apps. However, most of them have some disadvantage, such as low accuracy, not general, or not robust in big data. This paper proposes an automatic fake Apps detection method by using Decision Tree Ensembles based Detection (*i.e.*, DTED). The DTED is a three-level ensemble method which capitalizes different voting technology. We extract permission features and API calls features relevant to fake Apps behavior by analyzing a large number of samples. Experimental results show that our method achieves accuracy as high as 91.6% and 91.13% F-measure, which performs better than other algorithms in big data.

Introduction

With the rapid spread of smart phones, it has been more and more indispensable in people's daily life. Explosive growth of the software based on Android system has brought great convenience to people's lives. However, the number of fake Apps mixed in normal Apps is also growing at an alarming rate.

According to a recent study of the Japanese security software company, Trend Micro [1] has found more than 900,000 fake Apps on the trusted Google Play Store. These Apps are being used to steal your personal, financial information or aggressively serving ads. A survey of the top 50 free Apps available for download in Google Play, which conducted by Trend Micro company [3], revealed that almost 80% of the samples had fake versions. Fake Apps were more likely to the high-risk Apps or malware rather just mere harmless copycat [2]. As of April this year, of the 890,482 sample fake Apps discovered from various sources, 59,185 were detected ad aggressive adware and 394,263 were detected as malware. Among the fake Apps, more than 50% were deemed malicious [3].

In this paper, we propose an automatic fake Apps detection method by using Decision Tree Ensembles based Detection (*i.e.*, DTED), which is based on the power of ensemble learning. The DTED utilizes a decision tree ensemble to detect fake Apps from unknown Apps. The ensemble is built on specific three-level ensemble architecture and a novel combining method, which achieves not only a high accuracy but also a low false negative and false positive, which are represented by the Recall, Precision and F-Measure.

Related Work

As we focus on fake Apps in form of malicious code inserted, we mainly introduce some related works on mobile fake Apps which malicious code are inserted. Different from earlier work on mobile fake Apps depended on signature-based detection [4]. This type of approaches required high capacity repository for managing signatures, so it is not suitable for small mobile devices such as

Android and it causes deterioration of detection rate on new fake Apps since only predefined signatures can be detected.

Instead of using signatures for fake Apps detections, data mining and machine learning techniques provide an effective way to dynamically detect fake Apps through extracting fake Apps features [5,6]. Some existing works have used data mining, machine learning techniques, and features generated from Android .APK files. Schultz et al. In [7], Kolter et al. trained several machine learning algorithms on byte string n-grams. Sahs and Khan [8] trained a One-Class Support Vector Machine [9] based on features which were extracted from packaged Android applications (APKs). In [10], Zhou proposed a two-level ensemble architecture named Neural Ensemble based Detection (NED), which was well done in lung cancer cell identification but not in fake apps detection.

Motivated by the above methods, we combine permissions and API calls as features to characterize fake Apps, and propose the DTED method, which has a low false positive and false negative.

Methodology

In this section, we propose the DTED method, which utilizes permissions and API calls features to classify normal Apps and fake Apps. The framework consists of three major components in figure 1. The first component is an Android application analyzer, which decompresses the APK file and extracts AndroidManifest.xml and class file. The second component extracts permission features from AndroidManifest.xml file and API call features from class files. The third component is mainly about the DTED method.

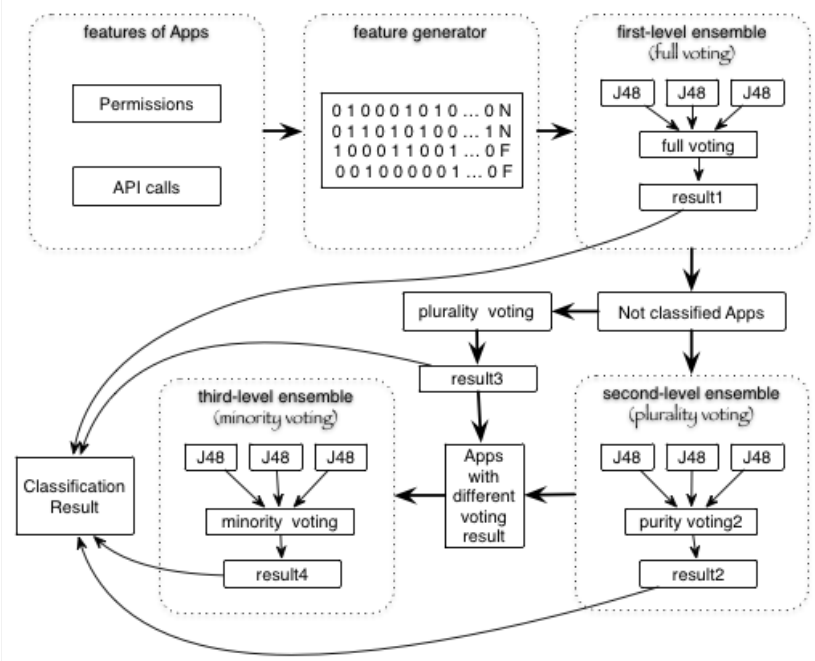

Fig. 1. A framework including feature generator and DTED method

After the processing of the first and second component, each app is represented as a single instance with permission and API call features, and a class label indicates whether the app is normal or fake. Since the second component has converted Apps into generic instance-features format, we can use the DTED method to classify unknown Apps. In the first-level ensemble, we utilize decision trees to classify unknown Apps, and use *full voting* method to get a classification: *result1*, i.e. all decision trees have voted *normal* or *fake*. All decision trees do not use all training data to train the decision tree model. Meanwhile, we utilize these decision trees and adopt *plurality voting* method to classify those Apps which were not classified through *full voting*. Through the steps above, we can get *result3*.

After the first-level ensemble, we can get some classified error instances as training data of the second-level ensemble learning. In the second-level ensemble, we can obtain different decision tree models through utilizing the same method with the first-level ensemble. The number of decision trees in the second-level ensemble is less than the number of decision trees in the first-level ensemble. In the second-level ensemble, we use *plurality voting* method to obtain the *result2*. Through combining the same results between *result2* and *result3*, we obtain the classification result in the second-level ensemble.

In the third-level ensemble, we put the classified error instances, which have different voting results between *result2* and *result3*, as the input of the third-level ensemble. However, the voting method in the third-level ensemble is different from the first-level and second-level ensemble. We utilize a novel *minority voting* method to get the *result4*. We combine the *result1*, *result2* and *result3*, *result4*, and get the final classification result.

Data Feature Structure

A. *Android Permission Sets*

In our dataset, the average number of permission requested by fake Apps is 13.99, while the average number requested by normal Apps is 7.95. These results show a fact that the frequency of requesting permissions between normal Apps and fake Apps has a huge difference.

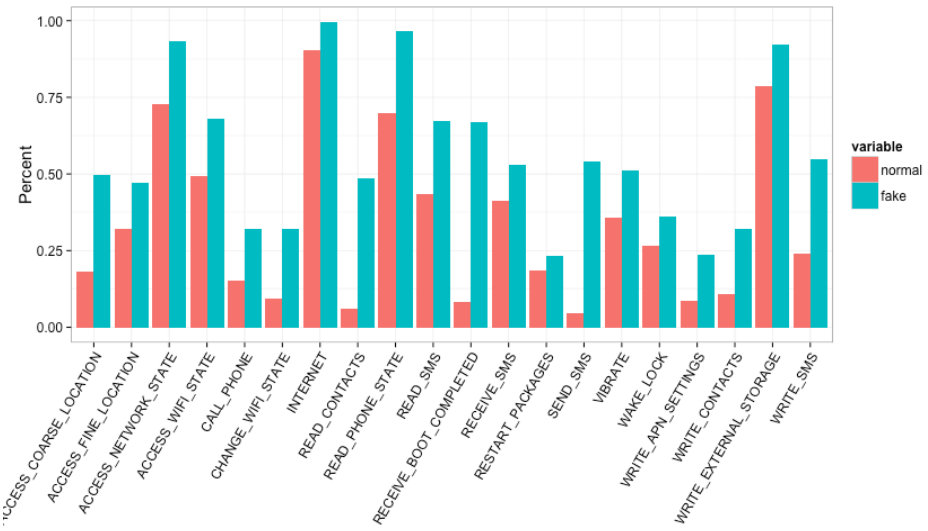

Fig.2. Comparison of top 20 requested permission by normal Apps and fake Apps

From above, we use the permission as the feature to characterize an app. We extract 55 permissions to characterize every app. As a result, every app is represented by a binary factor, namely P, where $P_i = 0$ if and only if the app has not requested the ith permission and $P_i = 1$ if and only if the app has requested the ith permission.

B. *Android API Calls*

In order to illustrating the reason vividly, we compare top 20 API calls used by fake Apps in our dataset with top 20 API calls used by normal Apps. The results are shown in Fig.3.

The same with characterizing permission, we use the API call as the feature to characterize an app. We extract 217 API calls to characterize every app. As a result, every app is represented by a binary factor, namely A, where $A_i = 0$ if and only if the app has not used the ith API call and $A_i = 1$ if and only if the app has used the ith API calls.

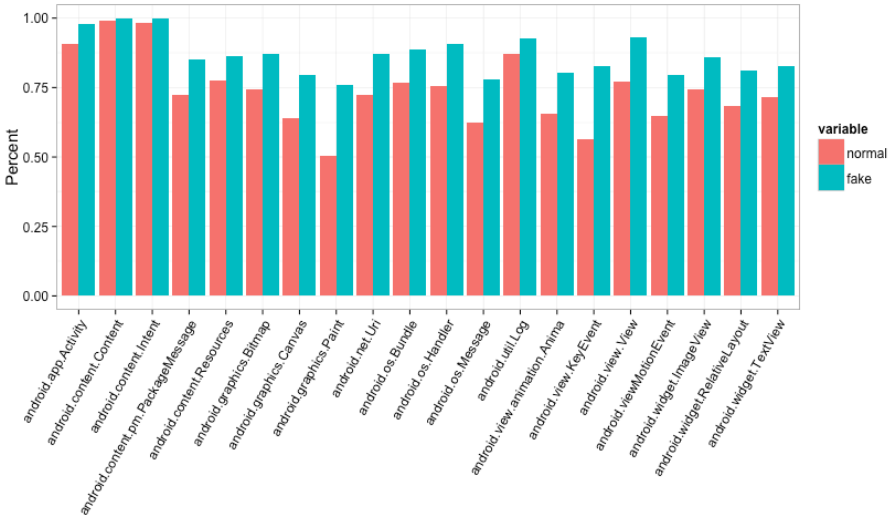

Fig. 3. Comparison of the top 20 used API by normal Apps and fake Apps

Evaluation and Experiment

A. *Data Set*

To extract fake Apps and normal Apps' permission and API call features, compare the DTED method with other classic machine learning algorithms, and evaluate the DTED method, we have collected and analyzed about 55,000 Apps. Our fake app sample consists of 26,160 fake Apps that we collected from different sources. The fake app sample spans a wide of categories in Google Play. Our normal sample consists of about 20,000 normal Apps, which are the top 500 free Apps in each category in Google Play that we collected in November 2014.

B. *Evaluation*

In order to show the superiority of DTED in classifying unknown Apps as either normal or fake Apps, we have compared the DTED method with three classic different machine learning algorithms: Random Forest [11,17], SVM [12] and J48 DT [13], a WEKA [14] implementation of the C4.5 algorithm [15]. These algorithms belong to different family of classifiers. J48 belong to decision trees and Random Forest is related to ensemble learning. SVM is a supervised learning method that proceeds through dividing the training data by an optimal separating hyperplane [16]. We try to show the prominent advantages of the DTED method in detecting fake Apps by comparison with other classification models. Through comparing and analyzing, we notice that DTED leads to a better accuracy compared to the other models, as shown in Fig. 4.

To test our classification models, we use split validation, which randomly splits our dataset into training data (70%) and testing data (30%). We train classification models based on the training data and test the classification models based on the testing data. To evaluate the classification models' performance, we adopt four different evaluation indexes: Recall, Precision, Accuracy and F-Measure. The definition of these indexes is showed as follows:

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

where *TP* is the number of fake Apps correctly classified and *FN* is the number of fake Apps classified as normal Apps. Recall is the percentage of accurately detecting fake Apps, and detection rate increases when high percentages are revealed. Similarly, Precision is the percentage of accurately predicting fake Apps in all predicting to fake Apps:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

where *TP* is the number of fake Apps correctly classified and *FP* is the number of normal Apps classified as fake Apps.

We can obtain F-measure with recall and precision:

$$F - measure = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{3}$$

Also, if we own the *TP, FP, FN* and *TN* values, we can get the Accuracy, which is the total number of fake and normal Apps correctly classified divided by the total number of the dataset instances:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

where *TN* is the number of normal Apps correctly classified.

Through adopting different evaluation indexes, we can find the advantage of DTED method in detecting fake Apps.

C. *Experimental Results and Analysis*

Fig. 4 shows that the experimental result of Accuracy, F-Measure, Recall and Precision through the four types of machine learning classification algorithms.

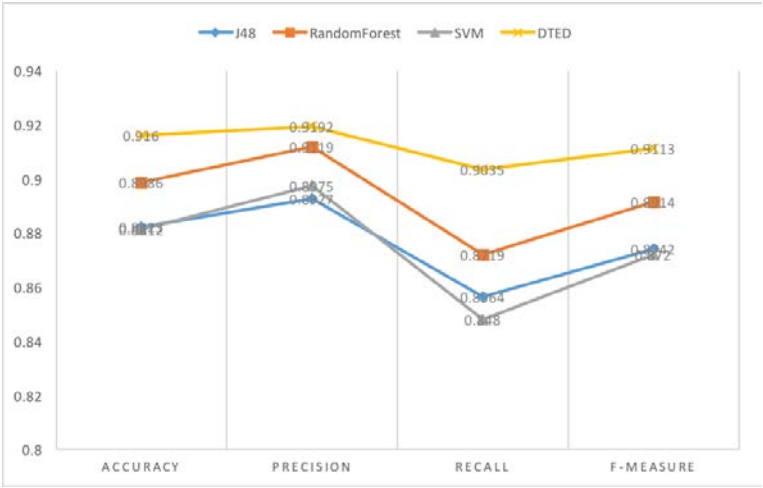

Fig. 4. Comparison of four classifier algorithms in Accuracy, F-Measure, Precision and Recall.

As seen in the graph, Recall and Precision of fake Apps type data greatly exceeded 75% in all machine learning algorithms. Also, Accuracy and F-Measure of all machine learning algorithms greatly exceeded 75%. The experimental result of the DTED method in Accuracy, F-Measure, Recall, Precision exceeded other machine learning algorithms in our experiments.

Conclusion

In this paper, we proposed and evaluated a new ensemble method for detecting Android fake Apps. This paper uses Permission and API call features by examining the structural features of Android APK file and utilizes the DTED method proposed to accurately detect fake Apps. The experimental results showed that the performance of the DTED was the most suitable in detecting fake Apps. The core of the DTED is a three- level ensemble architecture that not only comprises individual decision trees with different predictions but also utilizes different methods to combine individual predictions. In order to low the false negative and false positive, we utilize a novel prediction combining method named *minority voting* that is used in the third-level ensemble. Through adopting those techniques, DTED achieves not only high accuracy rate but also low false negative and false positive.

Acknowledgement

In this paper, the research was sponsored by National Natural Science Fund Project (Project No. 61175123) and Special Fund on Guangdong Province Chinese Academy of Sciences Comprehensive Strategic Cooperation (Project No. 2013B091300019).

References

[1] Trend Micro Incorporated. (2014). Trend Micro Mobile Threat Information Hub.” Malware in Apps’ Clothing: A Look at Repackaged Apps.” Last accessed May 27, 2014, http://aboutthreats. trendmicro.com/us/mobile/monthly-mobile-review/2014-04- malware-in-Apps-clothing.

[2] Trend Micro Incorporated. (October 2013). Trend Micro Mobile Threat Information Hub.” Malicious and High-Risk Android Apps Hit 1 Million: Where Do We Go from Here?” Last accessed June 30, 2014, http://aboutthreats.trendmicro.com/us/mobile/monthly-mobile-review/ 2013-10-malicious-and-high-risk-android-Apps-hit-1-million.

[3] S. Luo, P. Yan and Mobile Threat Research Team. (2015). A Trend Micro Research Paper.” Fake Apps: Feigning Legitimacy.” Last accessed March 22, 2015, http://www.trendmicro.com/ cloud-content/us/pdfs/security-intelligence/white-papers/wp-Fake-Apps.pdf.

[4] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, andromaly: A behavioral malware detection framework for android devices J. Intell. Inf. Syst., 38(1):161190, 2012.

[5] M. Schultz, E. Skkin, and E. Zadok. Data mining methods for detection of new malicious executables. In Security and Privacy Proceedings IEEE Symposium, pages 38-49, May 2001.

[6] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu. Virus detection using data mining techniques. In Proceedings of IEEE International Conference on Data Mining, 2013.

[7] J. Zico Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res., 7:2721-2744, December 2006.

[8] J. Sahs, and L. Khan. ”A Machine Learning Approach to Android Mal- ware Detection”, In 2012 European Intelligence and Security Informatics Conference (EISIC), pp. 141-147, 2012.

[9] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27, 2011. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

[10] Z. Zhou, Y. Jiang, Y. Yang, and S. Chen. Lung Cancer Cell Identification Based on Artificial Neural Network Ensembles. In Artificial Intelligence in Medicine, 2002, vol.24, no.1, pp.25-36.

[11] B., Leo.” Random Forests”. Machine Learning 45 (1):532.doi:10.1023/A:1010933404324.

[12] V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, NY, 1995.

[13] J. Quinlan, Introduction of decision trees. Machine learning, vol. 1, no.1, pp. 801106, 1986.

[14] S. Garner, Weka: The Waikato environment for knowledge analysis. In Proceedings of the 1995 New Zealand Computer Science Research.

[15] J. Quinlan, C4.5 programs for machine learning. Morgan Kaufmann Publishers, 1993.

[16] Aafer, Y., Du, W., and Yin, H. DroidAPIMiner: Mining API-Level features for robust malware detection in Android. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. eds. (2013) Security and Privacy in Communication Networks. Springer, Heidelberg, pp. 86-103.

[17] Wikipedia. Random Forest. [4-18-2015]. [http://en.wikipedia.org/wiki/Random_forest#cite note-breiman2001-1]