Identifying Latent Android Malware from Application's Description using LSTM

Zhiqiang Wu, Xin Chen, Scott Uk-Jin Lee
Department of Computer Science and Engineering, Hanyang University, Ansan, South Korea
E-mail: {wzq0515, xxtx0122, scottlee}@hanyang.ac.kr

Abstract—**Android is the most popular mobile system in the world where many applications provide convenient and diverse functions on top of it for our daily lives. However, a new Android malware is revealed every 10 seconds and the official application markets still consists of malicious and undetected applications due to the limitation of the existing malware detection techniques. In this paper, we propose an approach to identify the latent Android malware from application's description using Long Short-Term Memory (LSTM) technique. The actual permissions requested by source code and permissions predicted from the description using semantics analysis to are compared to verify the consistency. If an application requests a permission undeclared in the description or homogeneous applications, it will be considered as a latent Android malware. This approach can surely provide more secure environment for the end-users before they install the applications.**

Keywords-Android malware; Permissions; LSTM; Consistency; Word2Vec

I. INTRODUCTION

With the rapid advancement of mobile technologies, mobile applications provide diverse end-user functions that are very convenient to our daily life gradually making people to depend more on smart devices. Currently in Android, the most popular smart mobile device platform, number of malwares are ever increasing. Over 3.25 million Android malicious apps have been uncovered in 2016 indicating that a new Android malware is introduced every 10 seconds **[1]**. Although Google Play **[2]** verifies the security of uploaded applications, there are also possibilities of fishes escaping from the net. In addition, the end-users can also download applications from the third-parties store or repository where security of the application is not verified. The end-users without enough knowledge on security aspect of mobile applications cannot identify whether the downloaded application is malicious. These unverified and unreliable mobile applications may lead to the risk of devices hacking.

In order to build a secure Android platform for end-users, researchers and analysts have used diverse approaches to detect various Android malwares **[3-6]**. For example, DeepRefiner **[3]** utilizes the static analysis to explore malicious features based on XML file and bytecode in Android apps. However, this approach consumes a large amount of time for the detection. To reduce the time taken, Droid-Sec **[4]** applied a hybrid method to detect Android malware using deep learning. However, this approach generates a higher false positive rate. These related researches indicate that malicious Android applications cannot effectively be detected from source code and behaviors at runtime. Hence, the end-users are still valuable to download malicious applications from official or third-party markets.

Developers always put some amazing words in applications' descriptions to attract more end-users to download. These descriptions introduce the primary functions of the application and its novel features in details from which some specific permissions in the Android system required by the applications' functions and features can be determined. Therefore, we proposed a novel framework for detecting and verifying the consistency between actually requested permissions and permissions that are common or predicted from application's descriptions and its categories using Long Short-Term Memory (LSTM) technique **[7]**. We build a model to analyze and predict the possible requested permissions of the target application from its description in the download webpage. Then our approach will compare the predicted permissions with the actually requested permissions in *manifest.xml* after decompiling to identify whether the appliocation is malicious. The entire process of the proposed approach is completed before installing the application to avoid the possible threats from the unsecured applications.

The remainder of the paper is organized as follows. Section II describes the related works. Section III outlines the overall architecture of our approach and provide details of each step involved. Section IV discusses possible limitations and concludes the paper.

II. RELATED WORKS

Currently, there are many Android malware detection tools based on static and dynamic analysis **[8]**.

The static analysis approaches only analyze the source code to detect the malicious code and may lead to a large number of false positive as they cannot detect all possible cases of malware [1]. The dynamic analysis approaches on the other hand can detect the behaviors of applications

at runtime, but they require adequate input suites to sufficiently exercise execution paths.

Although the current researches have already achieved the high accuracy on malware detection, the existing tools cannot guarantee a secure environment for the end-users. There are few malicious applications still existing in the official market for download which infects end-users' mobile devices with a virus. Therefore, we propose an approach to avoid these malicious yet undetected applications before the end-users download.

III. METHODOLOGY

In this section, we first introduce the overall process of our approach as shown in Fig. 1. Initially, in order to extract more dataset as training data, we collect benign applications from Google Play. Then, the VirusTotal **[9]** with 51 anti-virus scanners are applied to check whether the collected applications are malicious. Each target application is considered as the training data only if all scanners show that the application is benign. If the target application is neither malicious nor benign, it is called a latent Android malware. Besides the benign applications, malicious applications will be obtained from VirusShare **[10]** which is a repository of malware samples provided to security researchers. The repository includes more than 32 million malicious samples where they will directly be taken as a malicious dataset.

Figure 1. Overall steps of the proposed approach.

A. Pre-processing

To detect the latent malicious applications, required permissions of the malicious dataset from VirusTotal and the benign dataset from VirusShare should be extracted. The procedure of such pre-processing is shown in Fig. 2.

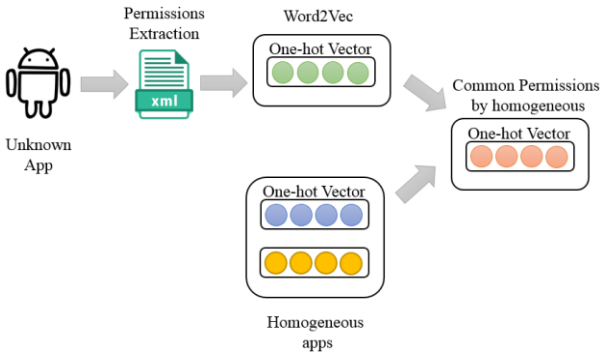

Figure 2. The procedure of pre-processing

Firstly, the Android apk files are decompiled to obtain source code for extracting actually requested permissions from manifest.xml file. The requested permissions in Android applications are represented in a one-hot vector using Word2Vec technique [7]. Then, the *n*k* matrix called Actual Permissions Matrix (APM) that consists of one-hot vectors representing the requested permissions from each application is created. In APM, *k* is the number of all usable permissions in Android system and *n* is the number of analyzed applications in the malicious and benign dataset. The commonly requested permissions for a category of applications can be extracted by the generated APM. For instance, social applications always request *INTERNET* permissions at runtime, which is common permission for social applications such as Facebook, WeChat and Kakao Talk. However, functions specific to an application are usually are not requested by other applications in the same category such as *LOCATION* permission.

$$\text{Proportion} = \frac{\#\ of\ apps\ used\ this\ permission}{\#\ of\ all\ apps\ in\ dataset} \quad (1)$$

$$\begin{cases} Proportion \geq theshold & 1 \\ Proportion < theshold & 0 \end{cases} \quad (2)$$

Hence, threshold is applied in the pre-processing stage to filter the most common permission in that category. If the proportion of the number of applications in Eq. (1) that request the target permission out of all applications in its category is more than the threshold, the permission will be considered common permission in this category of applications. Otherwise, the permission is considered as a specific to a function of an application that is different from others as shown in Eq. (2). If the permission is considered as a common in a category of applications, the value of the permission should be 1 in the index of the matrix called Common Permissions Matrix (CPM). CPM is an *m*k* dimension matrix where *m* is the number of application categories in Google Play. Other permissions are 0 in the CPM. Hence, each vector represents a set of permissions requested by a category of homogeneous applications.

B. Semantics Analysis and Prediction

In this step, stacked LSTM is applied to analyze the semantics of applications' description for predicting the latent requested permissions. The descriptions of applications on Google Play or other third-party markets are collected by crawler technique. The architecture of permissions prediction is shown in Fig. 3 below.

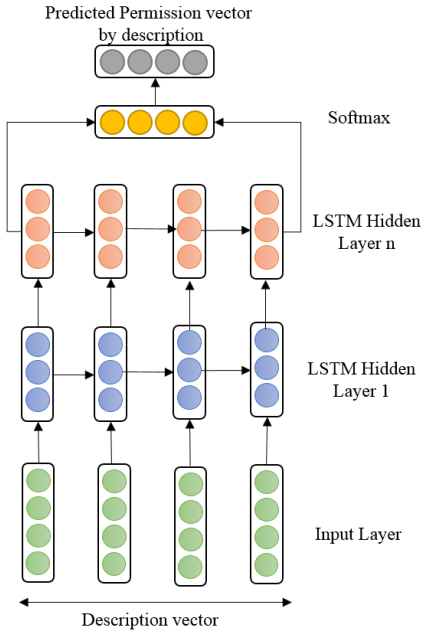

Figure 3. Permissions prediction by the description

The descriptions of applications usually introduce the attractive features and specific functions that are different

from other homogeneous applications. These application specific functions require some specific permissions. Therefore, we embed the descriptions that are natural language to a one-hot vector. Each sentence in the description will be an input data in this architecture. The stacked LSTM is then able to analyze each sentence that maps to a specific feature or operation in an application for predicting latent permission in Android. Stacked LSTM not only provides a predictor for each sentence, but also keeps the overall chrematistics which is not mentioned in the description for the whole application. For instance, as a social application, an application provides the services which depend on the Internet may not appear in the description because the Internet is basic permission required in social applications. The softmax is applied in the end of LSTM to identify the permissions requested in the application. Therefore, we will get a *k-dimension* one-hot vector as predicted requested permissions (PRP) for the target application.

C. Consistency Verification

Finally, the PRP is compared with the APM to analyze whether the target application requests the unknown permissions that were not mentioned in the description. Since some basic permissions will not be mentioned in the description such as *Internet* permission in social applications, *Location* permission in map applications and *Camera* permission in photo applications, CPM for different category is also considered as shown in Eq. (3).

$$\text{possible permission} = \text{CPM} \cup \text{PRP} \qquad (3)$$

The combination of CPM and PRP is called Possible Permission (PP) which includes all the latent requested permissions in the target application based on the description of the application and common permission in the homogeneous applications. Then, the PP is matched with APM as shown in Eq. (4) to get the Unknown Permissions Vector (UPV) which represents neither the common permissions in homogeneous applications nor the permissions identified by descriptions.

$$\text{unknown permissions} = \text{PP} \oplus \text{APM} \qquad (4)$$

If |UPV| is 0, it denotes that the actually requested permissions are same as its description and other homogeneous applications. Otherwise, the proposed approach will detect that the target application requests some permissions which is neither common nor predicted from the description and they will be refused in the installation part.

IV. Conclusion

In this paper, we present a novel approach to verify the consistency between the description of an Android application in the market and the actually requested permissions in *manifest.xml* of the application. The description usually denotes the functions specific to the target application. These specific functions may require some specific permissions which are not requested by homogeneous applications, which will be declared in its description to attract the end-users. We used LSTM to analyze the semantics of each sentence in the description for getting their latent permissions if the predicted permissions are exactly the same as actually requested permissions. We consider such application as a benign application which does not deliberately hide its function. Otherwise, the target application consists of latent risk due to the inconsistency between the predicted and actual requested permissions, which will be denied in the installation part. However, this approach can suffer from the cold-start issue. The end-users can not only download applications from the markets, but also share the applications with others. In this case, we cannot collect application's description for the detection. With the proposed approach, end-users can easily prevent using malicious applications, even the ones that are not identified by application markets, as they can be detected prior to actual installation.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. NRF-2016R1C1B2008624).

References

[1] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an and H. Ye, "Significant permission identification for machine-learning-based android malware detection," IEEE Trans.on Industrial Informatics, vol. 14, no. 7, pp. 3216-3225, July 2018

[2] Google Play: https://play.google.com/

[3] K. Xu, Y. Li, R. H. Deng and K. Chen, "DeepRefiner: multi-layer android malware detection system applying deep neural networks," in Proc. of IEEE European Symposium on Security and Privacy, pp. 473-487, 2018

[4] Z. Yuan, Y. Lu, Z. Wang and Y. Xue, "Droid-sec: deep learning in android malware detection," in Proc. of ACM Conference on SIGCOMM, pp. 371-372, 2014

[5] Y. Lee, J. Bang, G. Safi, A. Shahbazian, Y. Zhao and N. Medvidovic, "A SEALANT for inter-app security holes in android", in Proc. of IEEE/ACM International Conference on Software Engineering, pp. 312-323, 2017

[6] C. Gao, J. Zeng, M. R. Lyu and I. King, "Online app review analysis for indetifying emerging issues," in Proc. of IEEE/ACM International Conference on Software Engineering, pp. 48-48, 2018

[7] J. Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga and G. Toderici, "Beyond short snippets: deep networks for video classification," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 4694-4702, 2015

[8] M. Hammad, J. Garcia and S. Malek, "Self-protection of android systems from inter-component communcation attacks," in Proc. of IEEE/ACM International Conference on Automated Software Engineering, pp. 726-736, 2018

[9] VirusToTal: https://www.virustotal.com/#/home/upload

[10] VirusShare: https://virusshare.com/