2018 IEEE 14th International Colloquium on Signal Processing & its Applications (CSPA 2018), 9 -10 March 2018, Penang, Malaysia

One-Dimensional Convolutional Neural Networks for Android Malware Detection

Chihiro Hasegawa Hitoshi Iyatomi
Applied Informatics
Faculty of Science and Engineering
Hosei University, Japan
{chihiro.hasegawa.4w@stu., iyatomi@}hosei.ac.jp

***Abstract*—In recent years, malware aims at Android OS has been increasing due to its rapid popularization. Several studies have been conducted for automated malware detection with machine learning approach and reported promising performance. However, they require a large amount of computation when running on the client; typically mobile phone and/or similar devices. Thus, problems remain in terms of practicality. In this paper, we propose an accurate and light-weight Android malware detection method. Our method treats very limited part of raw APK (Android application package) file of the target as a short string and analyzes it with one-dimensional convolutional neural network (1-D CNN). We used two different datasets each consisting of 5,000 malwares and 2,000 goodwares. We confirmed our method using only the last 512-1K bytes of APK file achieved 95.40-97.04% in accuracy discriminating their malignancy under the 10-fold cross-validation strategy.**

***Index Terms*—malware identification; machine learning; 1D convolutional neural network**

I. Introduction

With the spread of smartphones and IoT techniques, the share of Google's Android OS is increasing rapidly [1]. Along with that, malware aiming at Android OS is also increasing, and new Android malware is found every day in the Android application market represented by Google Play [2]. Generally, there are three analysis methods to determine malware. They are surface analysis, dynamic analysis, and static analysis [3]. Surface analysis investigates character strings included in the target file. From the included character string, investigator might detect or obtain some meaningful information of the attack targets such as their domain, URL and so on. Dynamic analysis investigates the behavior of applications running on sandbox environment. This analysis is usually expensive, but makes it easier to find malicious communication etc. Static analysis is a method utilizing a code information obtained by disassembling and decompiling. For example, this analysis can see characteristics of malicious code such as using obfuscated variable names, functions etc. which make malware analysis difficult.

These are promising method, while they require expertise and experience especially on software development and binary analysis, thus they are fundamentally costly. In such backgrounds, several studies trying to detect Android malware have been proposed by using various machine learning techniques. Milosevic et al. [4] extracted permission information and source codes by decompiling of Android application package (APK) file from M0Droid dataset [5]. They analyzed them with typical machine learning techniques such as random forest, logistic regression, support vector machine, and their ensemble and they attained 0.956 in F-measure score. Yang et al. [6] converted character strings of several files (e.g. Dalvik executable file; executable on Dalvik virtual machine stored in APK file.) in the target APK file into image and analyzed it by using random forest with GIST features [7]. They evaluated their method using datasets from Android Drebin Project [8, 9] and their original method achieved 95.51% of precision on malignant detection task. Meanwhile, in the field of machine learning, studies on deep learning has gained considerable achievements among a very diverse range of problems, and thus is attracting attention. Under such circumstances, research that applies deep learning techniques to the security field has come to be seen [10].

Convolutional neural networks (CNN) [11, 12] is a representative machine learning method in deep learning, mainly used for image recognition problems. The most significant benefit of CNN over conventional approach is that it automatically obtains major features for the classification in the process of learning. This means that the system builder does not have to design and implement handmade features, those are the most essential but intractable part especially in large scale tasks. In natural language processing (NLP) task, character-level CNN (CLCNN) [13] or one-dimensional CNN (1-D CNN) has been used and showed excellent results [14]. CLCNN, as the name imply, is a CNN specially designed for text processing and it convolves character strings in a one-dimensional direction. The most significant advantage of CLCNN to mention is its high affinity for many effective and well-known strategies to improve the essential learning quality derived from CNN (e.g. data augmentation, drop-out for regularization, analyzing methodologies of feature maps for model interpretation, etc.). The next thing to mention is that the training time is much less than that of long short term memory (LSTM) [15] widely used in NLP tasks. In their application to malware detection, Huang et al. [16] converted Dalvik executable file into RGB color image and analyzed them with CNN. They attained 93% in accuracy on this two classes classification task. These image conversion methods are also seen in Windows malware detection in literatures [17-19].

978-1-5386-0388-8

These methods imaged executable file and performed the classification based on it. However, there is no basis to convert the target object, which is a one-dimensional data string, into an image. They are not theoretically sound and require additional hyper parameter (e.g. width of the image). Raff et al. [20] mentioned above issues and analyzed Windows portable executable (PE) as a character string for malware identification. They used a 1-D CNN and emphasized not to miss malicious codes that is unknown where they are embedded, and developed a method to convolve the entire file among the available memory constraints of the GPU. Their very shallow model uses very large kernel size of 500 and stride size of 500 for 2MB string inputs and achieves high classification accuracy of 94% by using global max pooling at the end. However, their calculation cost in training is tremendous; 16.75 hours / epoch with the latest eight GPUs, and therefore it is inferred that the time of execution at the client is also large. In sum, there is a concern that available situation might be limited. We therefore, propose a fast and accurate detection method for Android malware. Our method only uses the first or last N bytes code of raw APK file and investigates them with a 1-D CNN.

II. Method

APK is an archive file based on Java archive (JAR) format and, in fact, is a ZIP archive. It is composed of *classes.dex* (archive of executables on Dalvik virtual machine), *Android Manifest.xml* (permission information the application use), *res* directory (resource files) and so on. Generally, it is highly likely that these files contain malignant clues [21]. The actual location of these files stored in APK file depends on conditions and, of course, the location of malicious codes is unknown. Raff et al. [20] presented one possible solution for fundamentally the same issue as abovementioned. However, the size of the APK file is often several to 10 times larger than windows binary file. Therefore their methodology cannot be applied due to resource limitation in use in current situations as mentioned. We therefore proposed in this manuscript that we analyzed only the beginning or end N bytes of raw APK file with a 1-D CNN. The configuration of our 1D-CNN is shown in Fig. 1. and Table I.

All the convolution layers have (32 $\times$) 5 $\times$ 1 kernels, and pooling size in all maxpooling layer is 5 $\times$ 1. We applied batch normalization [22] in every convolution layer and rectified linear unit (ReLU) activation function is used in all layers. Finally, the output of final layer will be squashed with softmax function. For evaluation, we used the 10-fold cross-validation.

III. Experiments and Result

A. Dataset

We collected malware from two datasets; (1) AMD dataset [23] and (2) Android Drebin Project dataset. The former stores a total of 24,553 malwares and found 10 malware families have at least 500 samples. The latter contains 5,560 applications from 179 different malware families. These datasets only provide malware, so we needed to collect goodware dataset from other sources. We collected many APK files from Appsapk [24] and Apkpure [25], and inspected them with a total of 62 tools provided in VirusTotal [26]. We selected a total 2,000 applications as goodware identified as suspicious or malignant less than two tools out of 62.

In this experiment, we combined randomly choose a total 5,000 malwares from either of malware dataset (1) or (2) and 2,000 goodwares. We refer them as Dataset I and Dataset II, respectively. We also build a Dataset I (typical) that is composed of typical 10 types (500 $\times$ 10) of malwares from (1) AMD dataset and abovementioned 2,000 goodwares.

TABLE I
CONFIGURATION OF OUR SUPPOSED NETWORK

input ($N \times 1$)
Convolution (5×1)-32
Batch normalization
ReLU
Convolution(5×1)-32
Batch normalization
ReLU
Maxpool (5×1)
Convolution (5×1)-32
Batch normalization
ReLU
Convolution (5×1)-32
Batch normalization
ReLU
Maxpool (5×1)
Fully connect
Softmax

TABLE II
MALWARE DETECTION PERFORMANCE ON DATASET I

input size [byte]	accuracy (beginning) [%]	accuracy (end) [%]
512	94.29	95.98
1024	93.80	96.10
2048	92.04	96.51
4096	93.17	93.17

B. Experiments

Evaluation of malware detection performance was evaluated using the above datasets. In the experiments, we choose N = 512, 1024, 2048, and 4096 from the beginning and the end of the APK file to be analyzed, respectively. The results of identification performance for Dataset I, Dataset I (typical), and Dataset II are shown in Tables II, III, and IV, respectively.

IV. Discussion

Our experiments demonstrated our method attained very promising malware discrimination performance (maximum 97.04%) even with extremely short input data size N in all cases. We found no significant differences among discrimination performances for two datasets. More concretely, almost

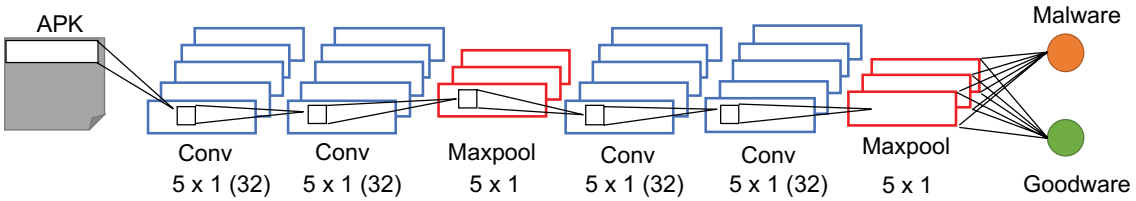

Fig. 1. Overview of our 1-D CNN model for malware discrimination.

TABLE III
MALWARE DETECTION PERFORMANCE ON DATASET I (TYPICAL)

input size [byte]	accuracy (beginning) [%]	accuracy (end) [%]
512	94.26	95.40
1024	92.68	96.48
2048	93.55	96.14
4096	93.22	94.00

TABLE IV
MALWARE DETECTION PERFORMANCE ON DATASET II

input size [byte]	accuracy (beginning) [%]	accuracy (end) [%]
512	95.67	96.36
1024	95.71	97.04
2048	94.89	94.68
4096	94.98	95.63

same performance in different dataset and different selection of data. For this reason, we believe the validity of this result and our method are high. In effectiveness of our method, since malware is treated as a short character string without being decompressed as it is, it can be executed with far less resources than conventional methods. Therefore, our method is considered to have great merit in actual operation. The analysis results selecting the N byte string from the back tended to be somewhat better than extracting it from the front for any dataset and for N size. Although they are not significant, it is considered that these areas contain more features that serve as clues for discrimination. We qualitatively examined the location of the files constituting APK file. Note here that APK file is composed of multiple files and compressed. However, we observed the position of constituting files in APK file considered to be important (such as *AndroidManifest.xml* which administers easy accessibility of human beings and *classes.dex* which is the executable file itself) is not necessarily included in the analysis N-length target. From the discrimination accuracy obtained, it is thought that our method captures some patterns that are difficult for human beings to directly interpret in the compressed files. In near future, we will investigate this in detail.

V. CONCLUSION

We proposed an accurate and practical light-weight malware detection method. Our method uses a shallow 1-D CNN and only analyzes a small portion of raw APK file without decompression. We confirmed from our experiments using plural datasets that malware can be identified with an accuracy of 96-97% on average by only analyzing the last 1 KB (N=1024) of the raw APK file. We assume our CNN appropriately captures some of key features of malware, while we could not confirm them at the moment because the APK file is a compressed and interpretation of string is intractable. We would like to analyze further in near future.

ACKNOWLEDGMENT

This research was partially supported by the Ministry of Education, Culture, Science and Technology of Japan (Grant in Aid for Fundamental research program (C), 17K8033, 2017-2020).

REFERENCES

[1] Operating System Market Share Worldwide — StatCounter Global Stats, http://gs.statcounter.com/os-market-share. [Accessed: 26- Nov- 2017].
[2] 8,400 new Android malware samples every day, https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day. [Accessed: 26- Nov- 2017].
[3] T. K. Barsiya, M. Gyanchandani, and R. Wadhwani,"ANDROID MALWARE ANALYSIS : A SURVEY PAPER," *International Journal of Control, Automation, Communication and Systems*, vol. 1, no. 1, pp. 35–42, January. 2016.
[4] N. Milosevic, A. Dehghantanha, and K-K. R. Choo,"Machine learning aided Android malware classification," *International Journal of Computers & Electrical Engineering*, vol. 6, pp. 266–274, July. 2017.
[5] M. Damshenas, A. Dehghantanha, K-K. R. Choo, and R. Mahmud,"M0Droid: An Android Behavioral-Based Malware Detection Model," *Journal of Information Privacy and Security*, vol. 11, no. 3, pp. 141–157, Sep. 2015.
[6] M. Yang and Q. Wen,"Detecting Android Malware by Applying Classification Techniques on Images Patterns," *International Conference on Cloud Computing and Big Data Analysis*, pp. 344–347, 2017.
[7] A. Oliva and A. Torralba,"Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.
[8] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket," in *Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS)*, 2014.
[9] M. Spreitzenbarth, F. Echtler, T. Schreck, F. C. Freling, and J.Hoffmann,"MobileSandbox: Looking Deeper into Android Applications," *28th International ACM Symposium on Applied Computing (SAC)*, pp. 252–276, March. 2013.
[10] H. Jiang, J. Nagra and P. Ahammad,"SoK: Applying Machine Learning in Security - A Survey," *CoRR*, abs/1611.03186v1, 2016.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,"Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, pp. 2278–2324, November. 1998.
[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.
[14] D. Shimada, R. Kotani, and H. Iyatomi, "Document classification through image-based character embedding and wildcard training," *IEEE Proc Big Data*, pp. 3922–3927, 2016.
[15] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-RecurrentNeural Networks," *CoRR*, abs/1611.01576, 2016.
[16] T. H. Huang and H. Kao, "R2-D2: ColoR-inspired Convolutional NeuRal Network (CNN)-based AndroiD Malware Detections," *CoRR*, abs/1705.04448v1, 2017.
[17] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath,"Malware Images: Visualization and Automatic Classification," *International Symposium on Visualization for Cyber Security*, no. 4, Jul. 2011.
[18] S. Yue, "Imbalanced Malware Images Classification: a CNN based Approach," *CoRR*, abs/1708.08042, 2017.
[19] F. C. C. Garcia, F. P. M. II, "Random Forest for Malware Classification," *CoRR*, abs/1609.07770, 2016.
[20] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware Detection by Eating a Whole EXE," *CoRR*, abs/1710.09435, 2017.
[21] M. Leeds, M. Keffeler, and T. Atkison, "A Comparison of Features for Android Malware Detection," *Proceeding ACM SE '17 Proceedings of the SouthEast Conference*, pp. 63–68, 2017.
[22] S. Ioffe, and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, abs/1502.03167, 2015.
[23] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep Ground Truth Analysis of Current Android Malware," *Detection of Intrusions and Malware & Vulnerability Assessment*, pp. 252–276, July. 2017.
[24] Appsapk, http://www.appsapk.com/. [Accessed: 26- Nov- 2017].
[25] Apkpure, https://apkpure.com/. [Accessed: 26- Nov- 2017].
[26] VirusTotal, https://www.virustotal.com/. [Accessed: 26- Nov- 2017].