Detection of Android Applications with Malicious Behavior Based on Sparse Bayesian Learning Algorithm

Ning Liu, Min Yang, Hang Zhang, Chen Yang, Yang Zhao, Jianchao Gan, and Shibin Zhang(✉)

The School of Cybersecurity,
Chengdu University of Information Technology, Chengdu, China
cuitzsb@cuit.edu.cn

Abstract. Android mobile devices are widely used in recent years. Due to the openness of Android, applications with malicious behavior have more opportunities to get confidential information, which can cause property damage. Most of current solutions are hard to detect these rapidly developing malicious applications with high accuracy. In this paper, a static malicious application detection method based on Sparse Bayesian Learning Algorithm and n-gram analysis is proposed to solve this problem.

Keywords: Malware detection · Android · N-gram
Sparse Bayesian Learning Algorithm · Dalvik opcode

1 Introduction

In recent years, the rapid development of mobile services help people can enjoy information services conveniently through mobile devices and mobile networks. A good example here is mobile payment, in the third quarter of 2017, Chinese mobile payment business reached 9.722 billion times and a total amount of 49.26 trillion yuan. However, malicious applications of the mobile device are also becoming more rampant. During the first quarter of 2017, a total of 58.127 million malwares were detected, which corresponds to 646,000 malware per day. Mobile devices need to improve their detection method to detect malwares and protect users' properties. Android is the most commonly used operating system of mobile devices. Openness is one of the most important reason for Android's success, but it also leads the system more vulnerable. As a result, detection of malicious applications is very critical for Android devices.

Two main types of malware detection methods are static analysis and dynamic analysis [1]. The difference between those types is that the static method will not run the detected software [2–5]. On the contrary, dynamic analysis method monitors the process when the application is running in the virtual machine, sandbox, even in the real environment [6–10]. Besides, some hybrid methods such as combined static and dynamic analyses, are also proposed to address the malware classifying problem [6, 11–13].

X. Sun et al. (Eds.): ICCCS 2018, LNCS 11067, pp. 266–275, 2018.
https://doi.org/10.1007/978-3-030-00018-9_24

1.1 Static Approaches

In [3], FlowDroid is proposed, which is a context, flow, field, object-sensitive and lifecycle-aware static taint analysis tool for Android applications. An SVM based method is introduced to classify Android malware in [4]. Enck et al. proposed Kirin, a malicious applications classifier, which checking the permission of Android applications [5].

1.2 Dynamic Approaches

A host-based Malware detection system, "Andromaly", is proposed, who monitors various features and events obtained from the Android device continuously in [7]. The malware is identified based on the monitored behavior. In [8], security specifications which are extracted from Manifest files of applications are compared to the data flows in ScanDroid.

1.3 N-Gram, SVM, and Sparse Bayesian Learning Algorithm

N-gram is always used for categorizing text [14]. In 2004 [15], N-gram has introduced in the malware detection filed. In [15] the proposed method is employed on bytecode. N-Gram has applied to analysis the bytecode of applications to extracted feature vectors, which combines SVM to classify the malware in [2]. However, the Training set was created based on known Benign/Malware application's features; it can only detect known vulnerabilities. Moskovitch et al. proposed another n-gram method which using opcode to substitute bytecode in the computer unknown malware detection [16]. More than 100 selected instruments opcodes, described by 7 or 10 symbols, are used to detect malware by MOSS (Measure Of Software Similarity) algorithm [17] and RF (Random Forest) [18], respectively.

SVM, the machine learning algorithm, is proposed in 1995 [19]. The aim is to find the hyperplane to classify samples. In Android malware detection, SVM and other machine learning algorisms are usually used to detect the malicious application depending on the source code [20].

Although SVM have shown state-of-art performance, it has some significant disadvantages [21]:

(1) Predictions of SVM are not probabilistic.
(2) The requisite number of support vector increase dramatically with the size of the training set.
(3) It is significant to select the error/margin trade-off parameter 'C'.
(4) The kernel function must satisfy Mercer's condition.

Tipping present Relevance Vector Machine (RVM) [21], which is a Bayesian treatment of the sparse learning problem. RVM offers probabilistic predictions and avoids the need to set parameters. SVM and RVM have both shown state-of-art results in classification problems [22]. However, the running time of RVM increased significantly as the amount of data. The running time of training algorithm scales approximately in the cube of the number of basis functions. An accelerated training algorithm

for RVM is presented in [23], which is named Fast Marginal Likelihood Maximisation for Sparse Bayesian Models. Since both of the RVM and the accelerated training algorithm are based on the Sparse Bayesian Models, they are also Sparse Bayesian Learning Algorithm (SBLA) [21].

In this paper, a combination of two algorithms, the n-gram analysis and SBLA, is developed, yielding a novel method that can further improve the efficiency of Android malware detection. The proposed method is compared with an SVM based algorithm for two scenarios. Results show that our method is effective and efficient for the detection of Android malware.

The paper is organized as follows: In Sect. 2 we describe our proposed method in detail. Section 3 delivers the configuration of our experiments. The results of experiments are discussed in Sect. 4. Finally, Sect. 5 concludes this article.

2 Our Method

A flow chart of our method is displayed in Fig. 1. The method will be explained in detail later in this section.

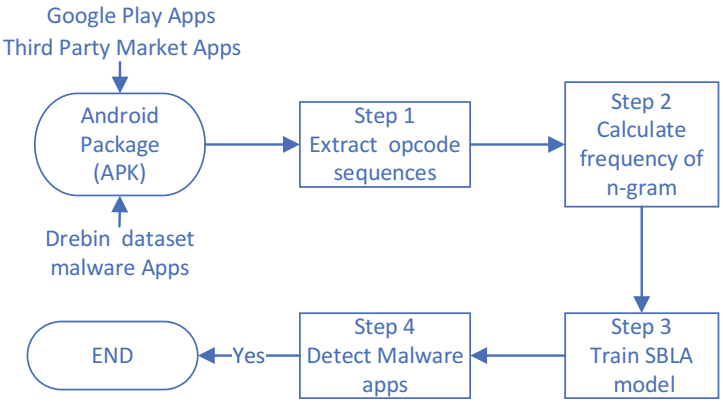

Fig. 1. Flowchart of our method

Android application can be delivered as an APK file, containing a manifest file, resource files and Dalvik executable files. The apktool can disassemble the smali file represents a single class that includes all the methods of the class. Each method contains Dalvik opcode and each instruction consists of a single opcode and multiple operands.

In our method, we disassemble the APK files of applications in step 1. Opcodes are extracted from each smali file. Since some Dalvik instructions of the applications are alterable when the applications are compiled in a different environment [24], those

alterable instructions are ignored when opcode are extracted. There are only seven core instructions sets are considered as follows:

(1) Move: which moves the content of one register into another one.
(2) Invoke: which is utilized to invoke a method
(3) accept one or more parameters.
(4) If: which is a jump conditioned by the verification of a truth predicate
(5) Return: which is used to return
(6) Goto: which jump unconditionally
(7) Aget: which gets an array element.
(8) Aput/ Iput: which put the integer value in into an array referenced.

For the sake of calculation, seven letters are used to describe the seven opcode sets in Table 1 [25].

Table 1. Letters and opcode name

Opcode	Move	Invoke	If	Return	Goto	Aget
Symbol	M	I	I	R	G	T

Secondly, in step 2, we calculate the frequency of n-gram in the opcode sequences of the applications. The output of this step is a vector of the n-gram from all the classes of the application. The vector contains the frequency of each n-gram [25].

Thirdly, in step 3, the accelerated RVM algorithm [23] is used to train the model.

Finally, in this step 4, the SBLA classifier is utilized for classifying malware. Accuracy, is defined as the ratio between the number of correctly classified samples and the number of all test samples, is selected to prove the effectiveness of the classifier:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

In (1), TP is True Positive, FP is False positive, TN is True Negative, and FN is False Negative. The result of a test sample always falls into one of those four basic categories.

3 Configuration and Scenarios

The benign Android applications are downloaded randomly from google app store and a third-party store (Tencent app store). The Android malicious applications are collected by the Drebin project [26]. There are 982 benign applications and 1114 malware applications.

There are two scenarios to prove the effectiveness and efficiency of the method.

(1) Scenario I: Prediction accuracy in different n-grams
Scenario I intends to obtain the highest accuracy model for different n-grams. In this scenario, we will test the productive of the combination of n-gram selected opcode analysis and SBL based algorithm. The influence of n-gram in accuracy and time consumption will be displayed. The ratio of the size of training samples set and that of testing samples is set as 80%.
(2) Scenario II: Prediction accuracy in the different number of training samples
Scenario II aims to obtain optimal prediction accuracy, under a different number of training samples. In this scenario, the influence of the number of training samples on the generalization performance of Sparse Bayesian Modeling Algorithm.

In this scenario, 3-gram is only considered. The detection results of an SVM based algorithm [25] and that of SBL based algorithm will be compared.

4 Results

In this section, the results of the scenarios described above are investigated for the proposed algorithm as well as for without and with genetic algorithms.

4.1 Scenario I: Accuracy and Time Consumption Between Different N-Gram Opcodes

The results of our proposed method working with different n-grams are presented in Table 2. It shows that 3-gram and 4-gram can get a fine classification accuracy. An interesting thing appearance is that when the sample dimension increases, the running time does not increase quickly.

Table 2. The results of scenarios I

N-gram	The accuracy of training samples [%]	The accuracy of testing samples [%]	Time consumption [s]
1-gram	83.44	83.49	2.7257
2-gram	92.68	91.93	13.4732
3-gram	96.11	94.96	16.9235
4-gram	97.31	95.51	18.2235

In Fig. 2, we can see that both methods can achieve an accuracy of 90% when the ratio of training samples is larger than 30%. However, the accuracy of SBLA is slightly lower than that of SVM with GA method.

In Fig. 3, it can be found that the SBLA-based method takes much less time consuming than the SVM-based method. As the number of training samples increases,

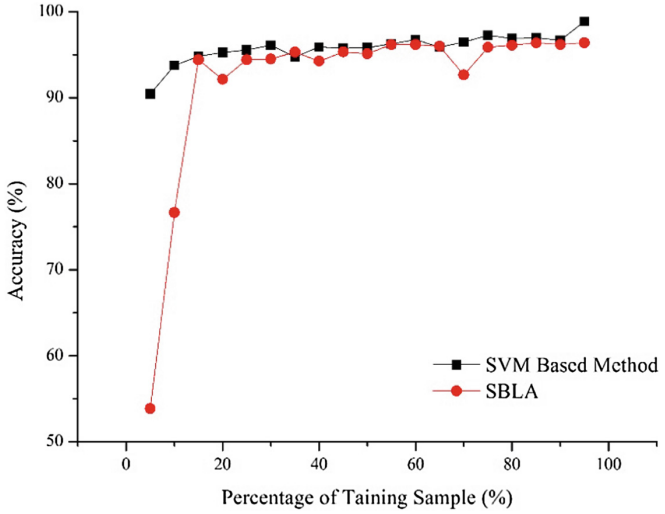

Fig. 2. The accuracy results of different algorithms.

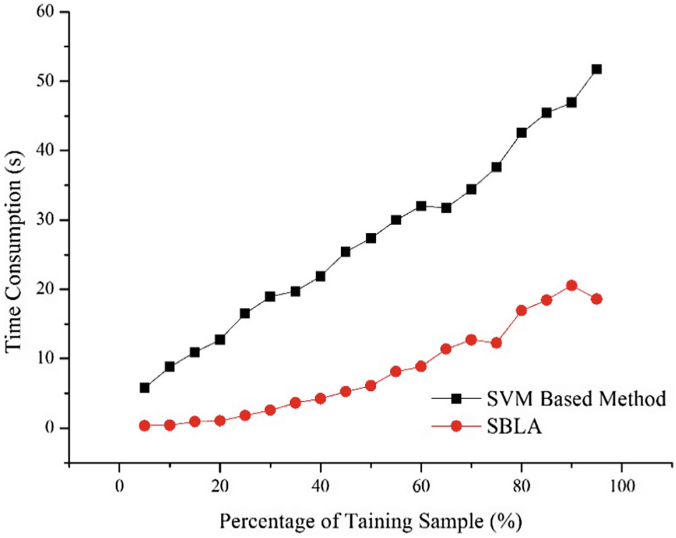

Fig. 3. The time consumption of different algorithms.

the SVM method exhibits a gradual increase in the time consuming, while the total cost of SBLA shows only a slight increase.

The accuracy of the SBL-based method is slightly lower than that of the SVM-based method. However, the time consumption of the SBL based method is significantly less than that of the SVM based method, especially while the training samples' number increase.

4.2 Scenario II: Accuracy in the Different Amount of Training Samples

The accuracy of the proposed method in the different ratio of the number of training samples to that of the overall samples is presented in Fig. 4. It shows that as the number of training samples reaches 300 (25% of overall samples), our method obtains results with fine prediction accuracy above 80%.

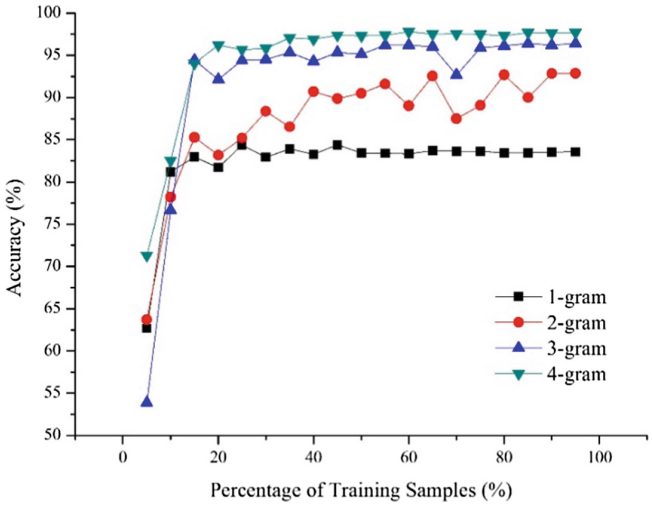

Fig. 4. Influence of the n-gram (1 to 4) in the accuracy.

When the number of training samples is above 300 (occupied more than 20% of overall samples), as the sequence grows, the overall classification accuracy of the algorithm is improved. However, when that of the training sample is below 200 (occupied lower than 10% of overall samples), the higher n of n-gram opcode the lower classification accuracy. This shows that long sequences require more training set samples to obtain better classification accuracy.

In Fig. 5, increases in the time consumption for the SBLA have been positively correlated with increasing number of training samples. However, the results also revealed less difference in time consumption between 2-gram and higher-dimensional samples, due to the sparsity of the SBLA [3].

Figure 6 shows the gap between the prediction accuracy of training samples and that of testing samples, under different training set numbers and n-gram opcodes. The higher number of samples' dimension, the higher gap of prediction accuracy obtained for the same number of training samples. Also, the algorithm has better generalization performance when the amount of training samples increases.

In general, n-gram analysis combined with the SBLA can effectively identify Android malware. Compared to the SVM based method, the SBLA achieves similar accuracy but saves time cost. Regarding n-gram analysis, 3-gram could already receive a great detection accuracy. There is no significant difference between 2-gram, 3-gram, and 4-gram when the number of samples is the same.

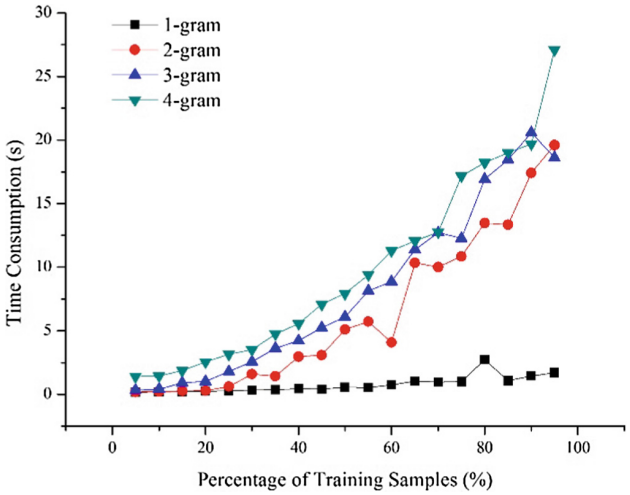

Fig. 5. Influence of the n-gram (1 to 4) in the time consumption.

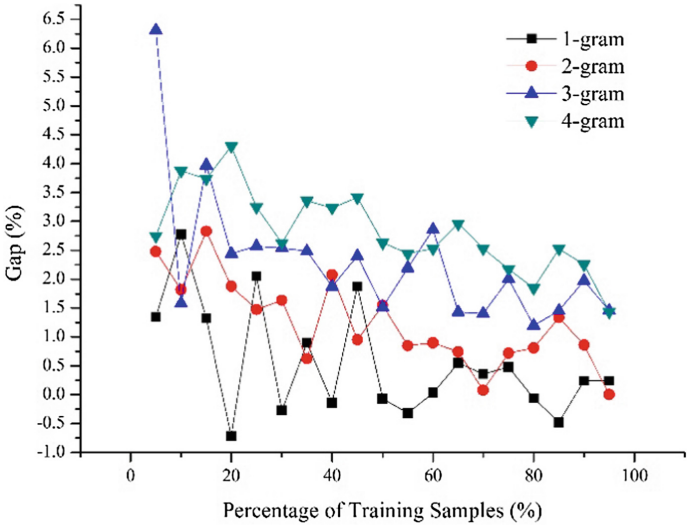

Fig. 6. Influence of the n-gram (1 to 4) in the accuracy.

5 Conclusions

We present a new method to detect applications with malicious behavior by extracting n-gram opcode from the applications and classifying malicious applications by SBLA. The experiments results indicate the effectiveness and efficiency of the proposed method. Future research will be extended to other anomaly recognition scenarios.

Acknowledgment. This work is supported by the National Key Research and Development Program (No. 2017YFB0802302), the National Natural Science Foundation of China (No. 61572086, No. 61402058), Sichuan innovation team of quantum security communication (No. 17TD0009), Sichuan academic and technical leaders training funding support projects (No. 201612008010264), Application Foundation Project of Sichuan Province of China (No. 2017JY0168).

References

1. Bergeron, J., Debbabi, M., Desharnais, J., Erhioui, M.M., Lavoie, Y., Tawbi, N.: Static detection of malicious code in executable programs. Int. J. Req. Eng. (2001)
2. Dhaya, R., Poongodi, M.: Detecting software vulnerabilities in android using static analysis. In: Proceedings of ICACCCT 2015, pp. 915–918 (2015)
3. Arzt, S., et al.: FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. ACM SIGPLAN Not. **49**(6), 259–269 (2014)
4. Li, W., Ge, J., Dai, G.: Detecting malware for android platform: an SVM-based approach. In: Proceedings of CSCloud 2016, pp. 464–469 (2016)
5. Enck, W., Ongtang, M., Mcdaniel, P.: On lightweight mobile phone application certification. In: Proceedings of CCS 2009, pp. 235–245 (2009)
6. Spreitzenbarth, M., Schreck, T., Echtler, F., Arp, D., Hoffmann, J.: Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques. Int. J. Inf. Secur. **14**(2), 141–153 (2015)
7. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y.: "Andromaly": a behavioral malware detection framework for android devices. J. Intell. Inf. Syst. **38**(1), 161–190 (2012)
8. Fuchs, A.P., Chaudhuri, A., Foster, J.S.: SCanDroid: automated security certification of Android applications (2010)
9. Enck, W., et al.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of OSDI 2010, pp. 393–407 (2010)
10. Yan, L.K., Yin, H.: DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In: USENIX Security Symposium, p. 29 (2013)
11. Patel, K., Buddadev, B.: Detection and mitigation of android malware through hybrid approach. In: Abawajy, Jemal H., Mukherjea, S., Thampi, Sabu M., Ruiz-Martínez, A. (eds.) SSCC 2015. CCIS, vol. 536, pp. 455–463. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22915-7_41
12. Faruki, P., et al.: Android security: a survey of issues, malware penetration, and defenses. IEEE Commun. Surv. Tutor. **17**(2), 998–1022 (2017)
13. Wen, W., Mei, R., Ning, G., Wang, L.: Malware detection technology analysis and applied research of android platform. J. Commun. **35**, 78–85 (2014)
14. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: 3rd Annual Symposium on Document Analysis and Information Retrieval, pp. 161–175 (1994)
15. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram-based detection of new malicious code. In: Proceedings of COMPSAC 2004, pp. 41–42. IEEE (2004)
16. Moskovitch, R., et al.: Unknown malcode detection using OPCODE representation. In: Ortiz-Arroyo, D., Larsen, H.L., Zeng, D.D., Hicks, D., Wagner, G. (eds.) EuroIsI 2008. LNCS, vol. 5376, pp. 204–215. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89900-6_21

17. Chen, T., Yang, Y., Bo, C.: Maldetect: an android malware detection system based on abstraction of Dalvik instructions. J. Comput. Res. Dev. **53**(10), 2299–2306 (2016)
18. Dong, H., Neng-Qiang, H.E., Ge, H.U., Qi, L.I., Zhang, M.: Malware detection method of android application based on simplification instructions. J. China Univ. Posts Telecommun. **21**(23–24), 94–100 (2014)
19. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
20. Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X.: On the automatic categorisation of Android applications. In: Proceedings of CCNC 2012, pp. 149–153 (2012)
21. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. JMLR.org (2001)
22. Ye, Y., Chen, L., Wang, D., Li, T., Jiang, Q., Zhao, M.: SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging. J. Comput. Virol. **5**(4), 283 (2009)
23. Tipping, M.E., Faul, A.C.: Fast marginal likelihood maximisation for sparse Bayesian models. In: Proceedings of AISTATS 2003, pp. 3–6 (2003)
24. Li, T., Dong, H., Yuan, C., Du, Y., Xu, G.: Description of Android malware feature based on Dalvik instructions. J. Comput. Res. Dev. **51**(7), 1458–1466 (2014)
25. Liu, N., Yang, M., Zhang, S.: Detecting applications with malicious behavior in Android device based on GA and SVM. In: Proceedings of ECAE 2018 (2018)
26. Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., Rieck, K.: DREBIN: effective and explainable detection of Android malware in your pocket. In: NDSS (2014)