ATLANTIS PRESS

Advances in Engineering Research, volume 140

2nd International Conference on Electrical, Control and Automation Engineering (ECAE 2017)

Detecting Applications with Malicious Behavior in Android Device Based on GA and SVM

Ning Liu, Min Yang and Shibin Zhang*

The School of Cybersecurity, Chengdu University of Information Technology, China

*Corresponding author

***Abstract*—In recent years, mobile technology and mobile-device have been rapidly developed. Since mobile devices collect and transmit large amounts of private information about users, malicious applications will pose a significant threat to the privacy and property security of the individual. Openness is a crucial factor why Android becomes the most popular mobile operate system, but it also results the Android system vulnerable to malware. In this paper, the n-gram opcode is employed to describe the applications, and then a static analysis method based on genetic algorithm and support vector machine is used to detect applications with malicious behaviors.**

Keywords—malware detection; Android; n-gram;* support vector machine*; dalvik opcode; genetic algorithm

I. INTRODUCTION

The rapid development of mobile Internet technology and mobile device has brought much convenience to people's life. The openness of Android platform has attracted most developers and users. Applications of Android platform provide effective services to user and collect huge private information of users. It also provides a new way for cyberattacks to make money illegally. During the whole year of 2016, users whose infected by programs with malicious behavior, the average number of malware infections was about 700,000 per day [1]. In terms of the number of malware, 360 Internet Security Center counted that there are 140.33 million novel malware samples for Android platform in 2016, with an average of 38,447 per day. As malware grows rapidly, antivirus needs to be updated more frequently and quickly to accommodate malware development.

In this paper, we propose a novel and efficient method to detect Android malware based on n-gram analysis and Support Vector Machine (SVM). Genetic Algorithm (GA), a heuristic algorithm, is used to improve the grid search to decrease time consumption of obtaining the optimal SVM model. The results indicate that the proposed method is able to effectively and efficiently classify Android malware.

The rest of this paper is as follows. Section II reviews previous work in detection of Android malware. Our proposed method is described in detail in section III, and the configuration and the results of experiments are discussed in Section IV and Section V. Finally, Section VI concludes this article.

II. RELATED WORK

Numerous research studies on the area of Android malware detection, especially in recent years. There are two types of malware detecting: static analysis and dynamic analysis [2]. The difference between those methods is that the detected software will not be ran in static method [3-6]. On the contrary, dynamic analysis method monitors the process when the application is running in the virtual machine, sandbox, even in the real environment [7-11]. Also, some hybrid methods, combined static and dynamic analyses, are also proposed addressing the malware classifying problem [7, 12-14].

A. Static Approaches

Flow Droid is a context, flow, field, object-sensitive and lifecycle-aware static taint analysis tool for Android applications[4]. Enck et al. proposed Kirin which classified the malicious applications by checking the permission of android applications [6]. In [5], an SVM based method is introduced to classify android malwares.

B. Dynamic Approaches

"Andromaly" is a host-based Malware detection system [8]. The system monitors various features and events obtained from the Android device continuously. In [9], security specifications which are extracted from Manifest files of application are compared to the data flows in ScanDroid.

C. Hybrid Approaches

Mobile-Sandbox, a static and dynamic analyzer combined with machine-learning algorithm, runs application in a sandbox for detecting malicious [7]. A hybrid approach, combined application analysis and generation algorithm, is proposed to classify the malware [12]. Both the static and dynamic analysis techniques are employed in this approach.

D. N-Gram, SVM and GA

N-gram is introduced in the malicious detection filed in 2004 [15], the method is employed on bytecode. In [3], N-Gram is introduced to analysis the byte code of application to extracted feature vectors, which combines SVM to classify the malware. However, the Training set was created based on known Benign/Malware application's features; it can only detect known vulnerabilities. In 2008, Moskovitch et al. proposed another n-gram method which using opcode to substitute bytecode in the computer unknown malcode 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 hyper plane to classify samples. In Android malware detection, SVM and other machine learning algorisms are usually use to detect the malicious application depending on the source code [20].

The penalty parameter C and kernel function parameters selection is crucial for SVM, since those have a heavy impact on the classification accuracy of SVM classifier [21]. The grid search is widely used for finding optimal kernel parameters [22], however it is time-consuming. GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes [23]. In [24], GA is introduced into SVM to find parameters for solving the bias problem and increasing the probability of correct classification of important samples.

III. ANDROID MALWARE DETECTION METHOD

A flow chart of proposed method is displayed in Figure I. The phases of the method will be explained in presented in following.

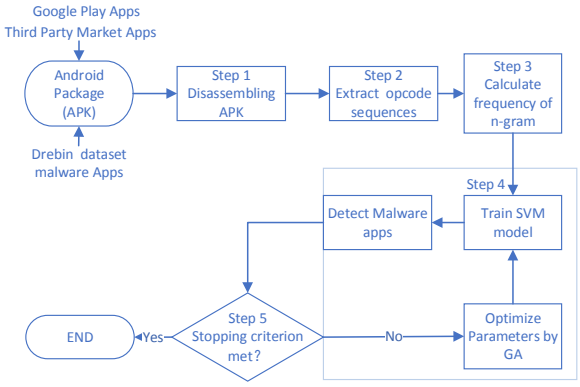

FIGURE I. FLOW CHART OF ANDROID MALWARE DETECTION

Firstly, in step 1, APK files of applications are disassembled. An Android application can be delivered as an APK file, containing a manifest file, resource files and Dalvik executable files. The apktool can disassemble the APK file and get the smali files of Dalvik instruction. Each smali file represents a single class that contains all the methods of the class. Each method contains Dalvik opcode and each instruction consists of a single opcode and multiple operands.

Secondly, in step 2, we extract opcodes from each smali file. Since some Dalvik instructions of the applications are alterable when the applications are compiled in different environment [25], those alterable instructions are ignored when we extract opcode. We only consider 7 core instructions sets as following:

1) Move: which moves the content of one register into another one

2) Invoke: which is utilized to invoke a method, it may

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, we choose seven letters to describe the seven opcode sets in Table I.

TABLE I. LETTERS AND OPCODE NAME

Opcode	Move	invoke	If	Return	Goto	Aget	Aput
Symbol	M	I	I	R	G	T	P

Thirdly, in step 3, 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.

The frequencies of n-gram are modeled as a vector which is defined as follows Table II.

TABLE II. AN EXAMPLE OF EXTRACTED N-GRAM FREQUENCY

n-gram	Vector	Frequency
MM	1	0.2%
MV	2	1%
MI	3	3%
…	…	…
PP	49	0.1%

$$V = 0.002, 0.01, 0.03\ldots 0.001\} \quad (1)$$

In (1), V represents a vector of 2-gram for an application.

Fourthly, in step 4, the SVM classifier, libSVM[6], is utilized for classifying malware. We select accuracy to measure the classifier. Accuracy is defined as the ratio between the number of correctly classified samples and the number of all test samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

In (2), 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.

GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes. The main driving operators of a GA are selection (to model survival of the fittest) and recombination through application of a crossover operator (to model reproduction).

GA is used to generate solution (parameter set), of which the best one will be selected founded on the evaluation of a fitness function corresponding to each solution. In our method,

the accuracy of SVM classifier trained by selected parameters is considered the fitness function value of the solution.

Finally, in this step 5, a stopping criterion is defined. The algorithm repeats step 4 until a maximum number of generations, generation size, is reached and returns the optimal solution as the best result.

IV. CONFIGURATION AND SCENARIOS

The benign android applications are randomly download from google app store and a third-party store (Tencent app store). The Android malicious behavior applications are collected by the Drebin project [26]. There are 650 benign applications and 661 malware applications are used for training. The test set contains 240 malware applications and 239 benign applications.

For SVM classifier, the RBF (Radial Basis Function) kernel is used to train the SVM model, since the number of vectors is limited. There are two significant parameters need to be selected: C (Penalty factor) and gamma (RBF kernel width). In general, the higher C leads the higher variance, but the lower bias. However, that is exactly the opposite of kernel parameter gamma. As gamma increases, the variance of the RBF kernel decreases.

The ranges of C and kernel parameter gamma in RBF kernel are both range from 1 to 200. In terms of GA, mutation and crossover possibility are both set as 0.5.

We consider three scenarios to prove the effectiveness and efficiency of the method.

1) Scenario I: Accuracy and time consumption between different n-grams without GA

Scenario I intends to obtain the highest accuracy model by grid search. In this scenario, we will test the productive of the combination of n-gram selected opcode analysis and SVM. We will also display the influence of n-gram between accuracy and time cost. The grid search is used to find the best parameter set. The grid size of 1 and 10 are considered.

2) Scenario II: Accuracy and time consumption between different n-grams with GA

Scenario II aims to obtain optimal accuracy model with acceptable time consumption. In this scenario, GA is used to improve grid search to find parameter set efficiently. The grid size of 1 and 10 are considered and different parameter of GA are also employed to show the influence. In this scenario, the size of population and generation are set as 10.

3) Scenario III: Influence of GA parameters.

In this scenario, 3-gram is only considered. The results with different population sizes and generation sizes will be display also to discuss their influence.

V. 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.

A. Scenario I: Accuracy and Time Consumption between Different N-Gram without GA

The comparison between the results of 1-gram, 2-gram and that of 3-gram for scenario I is list in Table 3. 2-gram and 3-gram all obtain a classifier with about 95% accuracy, showing the n-gram analysis and SVM work well. The accuracy of 3-gram is slightly higher than that of 2-gram, but it also costs a higher time consumption. Especially when the grid size is small, the grid search method is time-consuming.

TABLE III. THE RESULTS OF SCENARIOS I

n-gram	Grid Size	C	Gamma	Accuracy	Time consumption
1-gram	1	6	173	76.85%	4578 s
2-gram	1	53	112	94.4%	5348 s
3-gram	1	163	27	95.6%	28067 s
1-gram	10	10	100	76%	45 s
2-gram	10	60	110	94.2%	57 s
3-gram	10	200	30	95.6%	274.9 s

B. Scenario II: Accuracy and Time Consumption between Different N-Gram with GA

In this scenario, the average values of 20 times repeat experiments are presented as the results, for avoiding randomness influence of GA.

TABLE IV. THE RESULTS OF SCENARIOS II

n-gram	Grid size	Accuracy	Time consumption
1-gram	1	75.6%	7.3
2-gram	1	93.9%	9.3
3-gram	1	95.4%	46
1-gram	10	75.6%	7.8
2-gram	10	93.5%	9.7
3-gram	10	95.1%	47

When GA is added, the accuracy values slightly decrease around 1%, however the numbers of time consumption decline drastically above 60%.

C. Scenario III: Influence of GA Parameters

The average values of 20 times repeat experiments are presented as the results.

Figure II shows the influence of the population size in accuracy value and the time consumption. During the generation size and population size rising, the average values of time consumption almost increase uniformly. However, the accuracy is not increase obviously, after the population size reached 4.

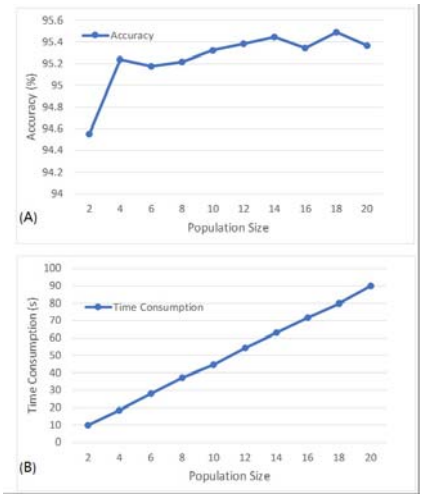

FIGURE II. INFLUENCE OF THE POPULATION SIZE (2 TO 20) (A) ACCURACY AND (B) THE NUMBERS OF TIME CONSUMPTIONS

Figure III indicates that the influence of the generation size is similar with that of the population size. When the generation size and population size are set 5 and 10 respectively, the method could get 95% accuracy values and only cost 20 seconds.

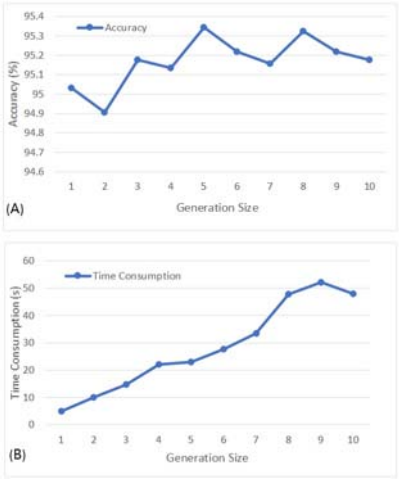

FIGURE III. INFLUENCE OF THE GENERATION SIZE (1 TO 10) ON (A) ACCURACY AND (B) THE NUMBERS OF TIME CONSUMPTIONS

VI. CONCLUSIONS

We have presented a novel method to detect malicious behavior applications by extracting n-gram opcode from the apps and using a hybrid algorithm to classify those applications. The experiments results indicate the effectiveness and efficiency of proposed method.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program (No. 2017YFB0802302), the National Natural Science Foundation of China (No.615720 86, No.61402058), Sichuan innovation team of quantum s ecurity communication (No.17TD0009), Sichuan academic and technical leaders training funding support projects (No. 201612008010264). Application Foundation Project o f Sichuan Province of China (No. 2017JY0168).

REFERENCES

[1] Center I.S.: '2016 China Mobile Security Status Report', in Editor: 'Book 2016 China Mobile Security Status Report' (2017, edn.), pp.

[2] Bergeron J., Debbabi M., Desharnais J., Erhioui M.M., Lavoie Y., and Tawbi N.: 'Static Detection of Malicious Code in Executable Programs', Int.j.of Req.eng, 2001

[3] Dhaya R., and Poongodi M.: 'Detecting software vulnerabilities in android using static analysis', in Editor: 'Book Detecting software vulnerabilities in android using static analysis' (2015, edn.), pp. 915-918

[4] Arzt S., Rasthofer S., Fritz C., Bodden E., Bartel A., Klein J., Traon Y.L., Octeau D., and Mcdaniel P.: 'FlowDroid:precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps', Acm Sigplan Notices, 2014, 49, (6), pp. 259-269

[5] Li W., Ge J., and Dai G.: 'Detecting Malware for Android Platform: An SVM-Based Approach', in Editor: 'Book Detecting Malware for Android Platform: An SVM-Based Approach' (2016, edn.), pp. 464-469

[6] Enck W., Ongtang M., and Mcdaniel P.: 'On lightweight mobile phone application certification', in Editor: 'Book On lightweight mobile phone application certification' (2009, edn.), pp. 235-245

[7] Spreitzenbarth M., Schreck T., Echtler F., Arp D., and Hoffmann J.: 'Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques', International Journal of Information Security, 2015, 14, (2), pp. 141-153

[8] Shabtai A., Kanonov U., Elovici Y., Glezer C., and Weiss Y.: '"Andromaly": a behavioral malware detection framework for android devices', Journal of Intelligent Information Systems, 2012, 38, (1), pp. 161-190

[9] Fuchs A.P., Chaudhuri A., and Foster J.S.: 'SCanDroid: Automated security certification of Android applications', 2010

[10] Enck W., Gilbert P., Chun B.G., Cox L.P., Jung J., Mcdaniel P., and Sheth A.N.: 'TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones', in Editor: 'Book TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones' (2010, edn.), pp. 393-407

[11] Yan L.K., and Yin H.: 'DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis', in Editor: 'Book DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis' (2013, edn.), pp. 29-29

[12] Patel K., and Buddadev B.: 'Detection and Mitigation of Android Malware Through Hybrid Approach' (Springer International Publishing, 2015. 2015)

[13] Faruki P., Bharmal A., Laxmi V., Ganmoor V., Gaur M.S., Conti M., and Rajarajan M.: 'Android Security: A Survey of Issues, Malware Penetration, and Defenses', IEEE Communications Surveys & Tutorials, 2017, 17, (2), pp. 998-1022

[14] Wen W., Mei R., Ning G., and Wang L.i.: 'Malware detection technology analysis and applied research of android platform', Journal on Communications, 2014

[15] Abou-Assaleh T., Cercone N., Keselj V., and Sweidan R.: 'N-gram-based detection of new malicious code', in Editor: 'Book N-gram-based detection of new malicious code' (IEEE, 2004, edn.), pp. 41-42

[16] Moskovitch R., Feher C., Tzachar N., Berger E., Gitelman M., Dolev S., and Elovici Y.: 'Unknown malcode detection using opcode representation', Intelligence and Security Informatics, 2008, pp. 204-215

[17] Chen T., Yang Y., and Bo C.: 'Maldetect:An Android Malware Detection System Based on Abstraction of Dalvik Instructions', Journal of Computer Research & Development, 2016, 53, (10), pp. 2299-2306

[18] Dong H., Neng-Qiang H.E., Ge H.U., Qi L.I., and Zhang M.: 'Malware detection method of android application based on simplification instructions', Journal of China Universities of Posts & Telecommunications, 2014, 21, (23-24), pp. 94-100

[19] Cortes C., and Vapnik V.: 'Support-vector networks', Machine learning, 1995, 20, (3), pp. 273-297

[20] Sanz B., Santos I., Laorden C., and Ugarte-Pedrero X.: 'On the automatic categorisation of android applications', in Editor: 'Book On the automatic categorisation of android applications' (2012, edn.), pp. 149-153

[21] Wang P., and Wang Y.S.: 'Malware Behavioural Detection and Vaccine Development by Using a Support Vector Model Classifier', Journal of Computer & System Sciences, 2015, 81, (6), pp. 1012-1026

[22] Hsu C.W., and Lin C.J.: 'A Simple Decomposition Method for Support Vector Machines' (Kluwer Academic Publishers, 2002. 2002)

[23] Holland J.H.: 'Adaptation in natural and artificial systems', Quarterly Review of Biology, 1975, 6, (2), pp. 126–137

[24] Liu S., Jia C.Y., and Ma H.: 'A new weighted support vector machine with GA-based parameter selection', in Editor: 'Book A new weighted support vector machine with GA-based parameter selection' (2005, edn.), pp. 4351-4355 Vol. 4357

[25] Li T., Dong H., Yuan C., Du Y., and Xu G.: 'Description of Android malware feature based on Dalvik instructions', Journal of Computer Research & Development, 2014, 51, (7), pp. 1458-1466

[26] Arp D., Spreitzenbarth M., Hübner M., Gascon H., and Rieck K.: 'DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket', in Editor: 'Book DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket' (2014, edn.), pp.