International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Reliable and Secure Classification Techniques for Spotting Malware in Mobile

Pooja B. Kote[1], S. M. Rokade[2]

[1]Department of Computer Engineering, SVIT, Chincholi

[2]Professor, Department of Computer Engineering, SVIT, Chincholi

Abstract: ***Nowadays there are many advanced techniques to hide from static and dynamic analysis tools in mobile. To get rid of this when attacking a mobile device an effective approach is required for the diagnosis of the application. In current approach to evaluate android app use of only simple code and pattern. The hacker can override this combination of diagnosis of pattern, as a result which may infect the device with the malware. This paper introduce approach which is using various techniques like patterns, flow based, behaviour based, state based and do analysis of each individual data by its associated specialized algorithms. The results obtained are fused to get the final results of that application. This paper aims to find malware using multi-classification technique. The algorithms will be used are Call Graph Based Classification, Neural network based Classification, and Naive Byes Based Classification. Experimental results show the feasibility and effectiveness of the proposed approach to detect the malware.***

Keywords: android, call graph, malware detection, naive byes, neural network

1. Introduction

Even if designing a malware is nowadays considered quite common the most advanced programmers try to hide malicious behaviours by using different techniques, such as the repackaging of legitimate applications or the obfuscation/ciphering of code. In current approach evaluates the android application for the detection of malware as it performs the analysis part by simple code or the pattern combinations. The hacker can override this combination of diagnosis of pattern, as a result which may infect the device with the malware.

In this promising approach, the system aims to spot malware using various combinations of algorithms and detect the malware. The algorithms that we are going to use are Call Graph Based Classification, NN based Classification, Naive-Byes Based Classification Experimental results show the feasibility and effectiveness of the proposed approach to detect the malware .To detect malware three approaches are used first approach require call graph technique to predict the calling relationship between subroutines. Second method is based NN classification i.e. back propagation algorithm and third method used is Bayesian classification represents a supervised learning method as well as a statistical method for classification.

The paper has three main contributions as follows:
1) Fusion of three algorithms.
2) To spot malware in android application.
3) Understand and evaluate various classification techniques.

In this paper, the approach aims to find malware in android application. Various classification technique such as Behavioral, pattern based, Call based. The remaining part of the paper is organized in different section as follows. The literature survey is given in section II. In section III the proposed technique with system architecture is explained. Conclusion is given in Section V.

2. Review of Literature

Karen A et al. [1] show a novel approach to host-based, post-mortem intrusion detection. In this post-mortem ID is of type anomaly, and is based on a classification method that combines a hidden Markov model (HMM) and *k*-means which we call KHMM.

To build a model for ordinary behaviour, in the first step, we factor out repetitive behaviour in a collection of ordinary (attack-free) log files. As a result, we obtain, first, a compressed version of all log files, and, second, a relation of the sequences of most frequent occurrence across all those logs, which we call across repetitive sequences. In the second step, we follow a 100-size, 100-step sliding-window approach to analyze every reduced log: starting at the first position of the log, we retrieve a window of size 100, then characterize each window by means of an attribute vector and then slide the window a step of 100 elements to continue with the same procedure in a third step, we build a model that captures the commonality in the sequence of attribute vectors, representing the original log.

Wei Wang, Xing Wang et al. [2] explains framework to identify malapps through analyzing the permission pattern usage, and app behavior is characterized by the permissions it requests.

Suleiman Y. Yerima et al. [3] gives perspective approach based on proactive method aimed at finding out known families as well as unknown malware so as to reduce malware that is present in android.

R. Andriatsim and efitra et al. [4] explain approach to detect an application infected with a malware, and look for a match between the information flows involving its data and the edges of the malware profiles.

V.Rastogi, Y. Chen et al. [5] this paper explained approach is to evaluate anti-malware products like an anti-virus for

Android regarding their resistance against various transformation techniques in known malware. For this purpose, approach developed Droid Chameleon, a systematic framework with various transformation techniques.

3. System Architecture

In this section, the brief discussion on proposed theory followed by proposed architecture.

The system takes android input as .apk file decompiles it. Once decompilation is done, then perform parsing to build data structure to give structural representation of input checking for correct syntax in the process. After this step, feature extraction which transforms the large data input into reduced data set called feature vector. Apply three classification techniques to this feature vector. Then classification and analysis of each one with different algorithms/methodology is to be made such as Behavioural analysis, neural network, Naïve-byes. Then collect result from all three and fused it. Finally output is malware found in mobile.

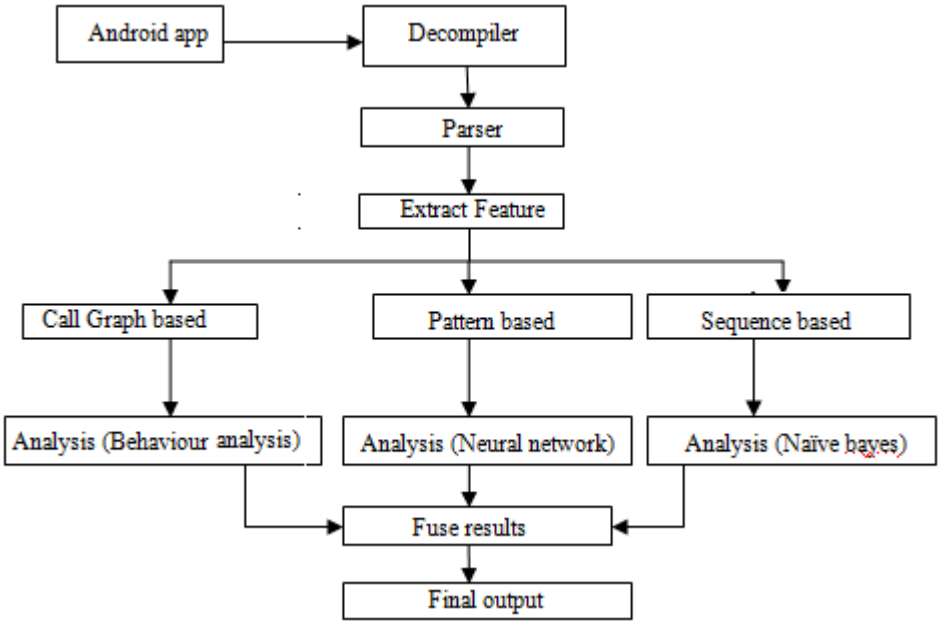

Figure 1: System Architecture

First phase-Decompilation it takes android app as an input and transforms it into code and decompiles it.

Second phase- Parser takes code as input and parsing build data structure, giving structural representation of input, checking for correct syntax in the process.

Fourth phase- Feature Extraction It transforms the large data input into reduced data set called feature vector.

Fifth phase –Classification
Call Graph based**:** In this technique will get graph representing relation between subroutines in code. Behavioural analysis will be used.

Pattern based**:** In this technique classification based on pattern matching .Neural network algorithm will be used.

Sequence based**:** In this technique sequence is transformed in feature vector and each sequence will have class label. Naive byes algorithm will be used.

Sixth phase- Analysis
The methods or algorithm will be applied to each of three techniques:

1) Behavioral analysis: It considers flow of function call and display function call graph.
2) Neural network: It is back propagation algorithm divided into two phases: Propagation and weight update.
3) Naïve byes: It is probability based algorithm. It consider feature vector and assign class label.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \qquad (1)$$

Seventh phase Fuse result: In this phase it will fuse result from all three techniques and will contain graph of recall and precision of each technique

Final output: Lastly final result that will give malware if present in app.

4. Result & Analysis

In this system we are detecting malware in android application in android platform. I am going to use malware dataset.

The performance measures used are:

1. Recall: Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were

not labelled as belonging to the positive class but should have been).

$$\text{Recall} = \frac{tp}{tp + fn} \quad (2)$$

2. Precision: The precision for a class is the number of true positives (i.e. the number of items correctly labelled as belonging to the positive class) divided by the total number of elements labelled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labelled as belonging to the class).

$$\text{Precision} = \frac{tp}{tp + fp} \quad (3)$$

3. Accuracy: Accuracy of classifier refers to the ability of classifier. It predict the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

Results of all techniques:

Table 1: Call graph based

Data Size	10	20	40
Precision	83.0	-	-
Recall	85.0	-	-

Table 2: Assembly based

Data Size	10	20	40
Precision	80.0	-	-
Recall	82.0	-	-

Table 3: Sequence based

Data Size	10	20	40
Precision	80.0	-	-
Recall	82.0	-	-

Analysis of Techniques

Table 4: Analysis of Techniques

Technique	Precision	Recall	Accuracy	False Positive	Total time
Call graph	83.0	85.0	84.0	-	-
Sequence based	80.0	820	86.0	1.4	0
Assembly based	80.0	82.0	79.0	2.1	0

5. Conclusion

This approach aims to provide an effective way, to spot malware in android application as three classification technique such as Call-graph, Naïve-byes and NN based is used. Fusion of all three techniques provides more efficient result. It is generous method can be applied to any android application. More reliable and portable as multi-classification is used to detect malware and can be used in any android device. The future enhancement is to add more additional feature or information to increase accuracy. The work can be extended to spot malware on direct application running on mobile device.

6. Acknowledgment

The author would like to thank the publisher and researcher for making their resources available. Also thanks to college who served as sounding support for both content and programming board. For their valuable and skilful guidance, assessment and suggestions from time to time improved quality of work in all respects.

References

[1] Karen A. Garc´ıa, Ra´ul Monroe, Luis A. Trejo, Carlos Mex-Perera, and Eduardo Aguirre, “Analyzing Log Files for Post mortem Intrusion Detection,” IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 42, no. 6, November 2012.
[2] Wei Wang, Xing Wang, “Exploring Permission-Induced Risk in Android for Malicious Application Detection,” Information forensics and security, vol. 9, no. 11, November 2014..
[3] Suleiman Y. Yerima, Sakir Sezer, Gavin McWilliams, “A New Android Malware Detection Approach Using Bayesian Classification” IEEE 27th International Conference on Advanced Information Networking and Applications, 2013.
[4] R. Andriatsim and efitra and V. V. T. Tong, “Detection and identification of Android malware based on information flow monitoring,” in Int. Conf. on Cyber Security and Cloud Computing, 2015, pp. 1–4.
[5] V.Rastogi, Y. Chen, and X. Jiang, “Catch me if you can: Evaluating Android anti- malware against transformation attacks,” IEEE Trans. On Information Forensics and Security, vol. 9, no. 1, pp. 99–108, Jan. 2014.
[6] Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “Andromaly: behavioural malware detection framework for Android devices,” Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161–190, 2012.
[7] Bose, X. Hu, K. G. Shin, and T. Park, “Behavioural detection of malware on mobile handsets,” in Proc. Int. Conf. on Mobile Systems, Applications, and Services, 2008, pp. 225–238.
[8] P. Faruki, V. Ganmoor, V.Laxmi, M. S.Gaur, and A. Bharmal,“AndroSimilar: robust statistical feature signature for Android malware detection,” in Proc. Int. Conf. on Security of Information and Networks,2013, pp. 152–159.
[9] W. Mazurczyk and L. Caviglione, “Information hiding as a challenge for malware detection,” Security & Privacy, vol. 13, no. 2, pp. 89–93, 2015.