Detection and Mitigation of Android Malware Through Hybrid Approach

Conference Paper · August 2015

DOI: 10.1007/978-3-319-22915-7_41

2 authors:

K. K. Patel
Charotar University of Science and Technology
14 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Bharat V Buddhadev
Malaviya National Institute of Technology Jaipur
10 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Detection and Mitigation of Android Malware through Hybrid Approach

Kanubhai Patel[1], Bharat Buddadev[2],

[1] CMPICA, Charotar University if Science & Technology, Changa, India
kkpatel7@gmail.com
[2] SS College of Engineering, Bhavnagar, India
bvbld@yahoo.com

Abstract. A good number of android applications are available in markets on the Internet. Among them a good number of applications are law quality apps (or malware) and therefore it is difficult for android users to decide whether particular application is malware or benign at installation time. In this paper, we propose a design of system to classify android applications into two classes i.e. malware or benign. We have used hybrid approach by combining application analysis and machine learning technique to classify the applications. Application analysis is performed by both static and live analysis techniques. Genetic algorithm based machine learning technique is used to create rules for creating rule base for the system. The system is tested with applications collected from the various markets on the Internet and two datasets. We have obtained 96.43% detection rate to classify the applications.

Keywords: Malware analysis, instrumentation, rule-based learning, classification, machine learning.

1 Introduction

The number of mobile and smart phones' users has significantly increased in last few years. Android is the most predominant and popular platform for mobile and smart phone in the world. Because of this popularity, a good number of android applications are available in various markets on the Internet. More than 1,500,000 apps are available on Google Play Store alone. Among these a good number of applications are law quality apps (or malware). It is difficult for android users to decide whether particular application is malware or benign during installation time.

Mobile and Smart phones' users store their private and security related information in the devices. This results into information stealing through malwares and also misusing of it. There are many other side-effects of which most naïve users are unaware of.

We propose a design of system to classify android applications into two classes i.e. malware or benign. We have used hybrid approach by combining application analysis and machine learning technique to classify the applications. Application analysis is performed by both static and dynamic (or live) analysis techniques. Also machine learning technique is used to create rules for creating rule base for the system. We

have obtained 96.43% detection rate to classify the applications. Also we have identified few weak points in our design that we need to rectify for better performance.

The remaining of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the design of our system. Section 4 describes experimental study and results. Section 5 concludes our paper along with future directions.

2 Related Work

There are two types of Android application analysis viz., i) static analysis, and ii) dynamic (or live) analysis. A few researchers have used machine learning techniques to classify the apps into benign or malware [1] along with static analysis and dynamic analysis of apps.

2.1 Static Analysis

Few researchers have used static analysis approach alone for android [2] [3] [4] [5] [6] [7] [8] [9]. Schmidt et al [2] use static analysis approach for classification by using readelf command. To improve the performance of Android malware detection, Kang, Jang, Mohaisen, & Kim [3] use the creator's information, like serial number of certificate, as a feature to classify malicious applications. They use similarity scoring to increase detection accuracy.

Ded [4], CHEX [5], AppSealer [6], FlowDroid [9], Capper [7], and PEG [8] use static analysis of data flow to detect malicious code in Android applications.

2.2 Dynamic Analysis

Andro-profiler [10] and Crowdroid [11] perform system call classification based on dynamic analysis. Mulliner, Oberheide, Robertson, & Kirda [12] propose PatchDroid. It uses dynamic binary instrumentation to inject patches dynamically in Android application. Zhou, Wang, Zhou, & Jiang [13] proposed DroidRanger.

Grace et al [14] propose an automated system called Risk-Ranker to analyze behaviors to detect root exploits. Pandita et al [15] propose Natural Language Processing (NLP) based system called WHYPER TaintDroid [16], DroidScope [17], and VetDroid [18] have conducted dynamic taint-analysis to identify malicious code. Yan and Yin [17] present DroidScope, an android analysis platform that uses virtualization-based malware analysis.

2.3 Machine Learning Based Detection

A good number of reserchers have used machine learning techniques to classify Android malware [11] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] automatically. Crowdroid [11] identifies trojan on Android smart phones using

machine learning approach. Andromaly [19] uses behavior-based approach. MADAM [20] provides both kernal level and user level android malware detection using 13 features.

H. Peng et al. [26] classify permissions of android apps. Schmidt, et al. [21] extract features by monitoring smart phones. They detect anomalies using these features with machine learning algorithm. Enck, Ongtang, & McDaniel [23] and Ongtang, McLaughlin, Enck, & McDaniel [24] present Kirin security service, while Xie, Zhang, Seifert, & Zhu [22] propose pBMDS for Android smart phone.

Juxtapp [27] implemented feature hashing on the operation code sequence. Zhang, Duan, Yin, & Zhao [25] present DroidSIFT. DroidAPIMiner [28] use feature extraction at the API level. It also provides light-weight classifiers. While DREBIN [29] use feature extraction by considering both Android permissions and API calls.

3 Design

Our design uses static and dynamic (or live) analysis to analyze the apk and machine learning approach to derive rules to classify android apps into two classes. Fig.1 shows schematic overview of design of our system.

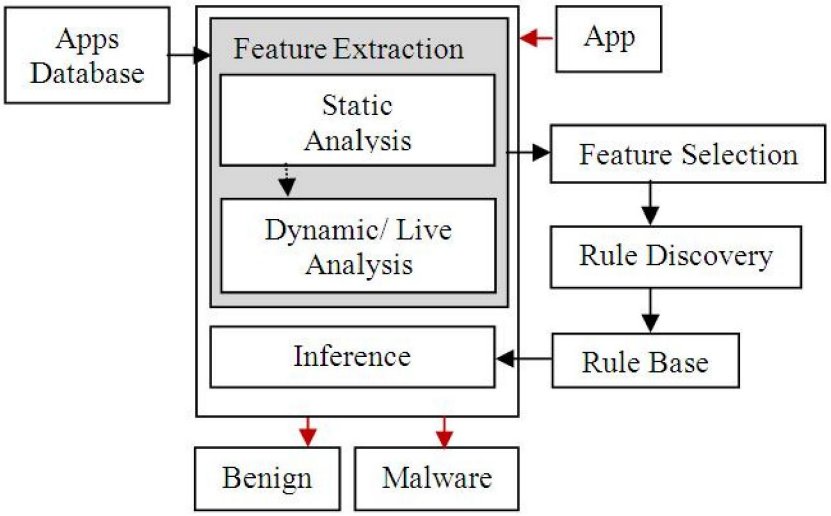

Fig. 1 Schematic overview of our system components

3.1 Feature Extraction

This module uses android application database and carry out on each apk serially. It converts apk to zip file and then unpack zip into different folders. Folder name is same as apk name. During static analysis, the module scans the manifest file of each

apk. From this manifest file it generates comma separated values (CSV) file with various permission-based parameters. We use Android Asset Packaging Tool (aapt) which is part of the Android SDK.

More than 125 features are extracted related to permission of apk (android.permission). Some of the extracted features are as below:

ACCESS_WIFI_STATE	READ_OWNER_DATA	SEND_SMS
CHANGE_WIFI_STATE	READ_CONTACTS	RECEIVE_SMS
	READ_PHONE_STATE	CALL_PHONE
ACCESS_NETWORK_STATE	READ_SMS	VIBRATE
CHANGE_CONFIGURATION		WAKE_LOCK
	GET_ACCOUNTS	
MODIFY_PHONE_STATE	WRITE_EXTERNAL_STORAGE	NFC
MODIFY_AUDIO_SETTINGS	WRITE_SECURE_SETTINGS	INTERNET

Also, we extract information regarding intents, services, and broadcast receivers from the manifest file. After this, the classes.dex file is converted to smali [30] to get information regarding functions and methods. We are interested in the functions and methods which are commonly seen in malwares. For example, to get the country name of the smart phone user, apps are call getSimCountryIso(). Some of other functions and methods information retrieved from this smali are as below:

Name of methods/Functions	Purposes of methods/Functions
getPackageInfo()	To search for installed AV products
sendTextMessage()	To send SMS messages
Ljava/lang/Runtime;->exec()	To execute the particular command in a separate process. Malwares generally call su and chmod commands.
encryption libraries functions	To find use of encryption and obfuscation within the applications.

A partial CSV file is created as a result of the static analysis. Many kind of malicious behaviours can be observed during run time only. For this type of runtime behaviour observation, apps are executed on the Android emulator [31] and interact with them. Mainly we observed higher-level language (java) based calls, lower-level language (native code) based functions calls, and user intraction. Also we capture the network traffic.

Java code based calls are mainly for runtime activities of the app. This includes data accessed by the app, data written to the files, sending SMS, data sent to the networks and data received over the network etc. Lower-level language (native code) based functions calls are observed through ltrace [32] command. Some of the native code functions are mainly used for communication purposes such as connect(), socket(), read(), and write() functions.

For some of malwares, user's interactions are required to interact with app. User interaction with application requires command passing to the application through touch screen. For this purpose we use MonkeyRunner toolkit [33] which is available in Android SDK. We trigger every interaction events at least once. Lastly, we capture the network traffic by using tcpdump. As a result of the dynamic analysis a full CSV file is generated. More than 210 features are extracted. To improve the performance

of rule discovery module we select the most desirable features only by feature selection module.

3.2 Feature Selection

We apply information gain technique to select most effective features only. This feature selection process make it possible to select only approprite features which avoids over-fitting, reducing generality, misleading the learning algorithm, increasing model complexity, and descreasing run-time and there by it improves the performance. It is more needed in smart phones where there is resources constraints.

We use a normalized information gain, also known as asymmetric dependency coefficient (ADC), for estimation quality of each feature. ADC [34] is specified as

$$ADC(C,f) = \frac{M\,I\,(C,f)}{H(C)} \quad (1)$$

Where H(C) is class entropy, and H(f) is feature entropy. MI(C, f) is mutual information between class C and feature f. This is defined by Shanonna [35] as:

$$H(C) = -\sum_{i=1}^{C} p(Ci) lg_2 p(Ci)$$

$$H(f) = -\sum_{x} p(f = x) lg_2 p(f = x)$$

$$MI(C,f) = -H(C,f) + H(C) + H(f) \quad (2)$$

CSV file is generated by feature extraction and selection module. This CSV file contains permission related information alomg with behaviour related information of the app as well as class of the app i.e. ‘Malware’ or ‘Benign’.

3.3 Rules Generation

Rule discovery module uses above CSV files and genetic algorithm to derive predictive rules to classify the applications into two classes i.e. Malware or Benign. It generates IF-THEN rules. We have used approach as described by us in Patel and Buddhadev [36]. Some of the rules generated are as below:

If apk does not have permission to send SMS message in manifest file and that apk try to send SMS messages then that apk may have malicious code.

IF Not (SEND_SMS) && CALL_sendTextMessage THEN Malware

If apk does not have permission to call (CALL_PHONE) in manifest file and that apk try to call on any premium number then that apl may have malicious code.

IF Not (CALL_PHONE) && Intent.ACTION_CALL THEN Malware

4 Experimental Results

To verify the design of the system we have implemented the system and carried out experiments.

4.1 Dataset Description

We have collected around 755 Android applications from various sources. Also we have used DroidKin [37] and ContagioDump [38] datasets. We have created a dataset from these android applications by extracting features. More than 251 features are extracted related to behaviour and permission of apk.

4.2 Implementation

The proposed technique is implemented in software using Java language. We have created classes for feature_extraction, feature_selection, individual, population, rule_evaluator (for fitness calculation), and breeder (for crossover and mutation). Feature_extraction module loads the apk one after other and scans it. Dynamic analysis loads the apk in to Google emulator and run the apk. Various commands are passed through the system to the apk to check the behavior of the apk. As a result of static analysis and dynamic analysis various features are extracted.

Feature_selection module use information gain technique and select the most appropriate features for rule discover. This results in to improvement of performance of the system which is desirable in mobile computing. Individual class describes chromosome representation and fitness value. Population class initializes the population. Rule_evaluator class is for to calculate the fitness using the the fitness function. Breeder class is for performing crossover and mutation to generate new population. In first phase, application uses above dataset to generate rule base. The system was trained by setting the parameters as below:
No of generations: 500
Cross over: one-point cross over
Probability of crossover: 0.7
Mutation rate: 0.01

In second phase, we test the system using this rule base and the testing datasets creating by us.

4.3 Results

This section presents the results of the experiments and system evaluation. We evaluated the system based on detection rate and performance of the system. In the testing phase, we have obtained 96.43% detection rate as shown in Table-1. There is scope of improvement in term of the performance of the system, as it takes around 8 to 11 minutes to scan the application and extract the features of the single application alone.

Table 1. Performance of the system

Parameter	Value
Detection rate	96.43%
Scanning time	8 to 11 minutes

5 Conclusion and Future Works

We attempt to explore the use of hybrid approach for classifying android applications into malware or benign in this paper. We use android application (static and live) analysis along with machine learning technique to extract the features and discover the classification rules. The proposed system is tested on collected android applications from open markets as well as DroidKin [37] and ContagioDump [38] malware datasets. Our system gives 96.43% detection rate during testing phase. The weak point of our system is that it takes 8 to 11 minutes to scan the application. As future work, we would like to improve the performance of the system by performing various operations in parallel.

Acknowledgments. We are thankful to the management of Charotar University of Science & Technology for providing support for the research. Special thanks to Dr. Ajay Parikh and Dr. S K Vij for their support and help.

References

1. Spreitzenbarth, M., Schreck, T., Echtler, F., Arp, D., Hoffmann, J.: Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques. International Journal of Information Security, 14(2), pp. 141-153 (April 2015).
2. Schmidt, A., Bye, R., Schmidt, H., Clausen, J., Kiraz, O., Yuksel, K., Camtepe, S., Albayrak, S.: Static Analysis of Executables for Collaborative Malware Detection on Android. In : IEEE International Conference on Communications, 2009 (ICC '09), Dresden, pp.1-5 (2009)
3. Kang, H., Jang, J.-w., Mohaisen, A., Kim, H.: Detecting and Classifying Android Malware Using Static Analysis along with Creator Information. International Journal of Distributed Sensor Networks, 2015, Article ID 479174, 9 pages, (2015).
4. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A study of android application security. In : 20th USENIX conference on Security (SEC'11), pp.21-21 (2011)
5. Lu, L., Li, Z., Wu, Z., Lee, W., Jiang, G.: CHEX: statically vetting Android apps for component hijacking vulnerabilities. In : 2012 ACM conference on Computer and communications security (CCS'12), pp.229-240 (2012)
6. Zhang, M., Yin, H.: AppSealer: Automatic Generation of Vulnerability-Specific Patches for Preventing Component Hijacking Attacks in Android Applications. In : 21st Annual Network & Distributed System Security Symposium (NDSS'14), San Diego, CA (2014)
7. Zhang, M., Yin, H.: Efficient, Context-aware Privacy Leakage Confinement for Android Applications Without Firmware Modding. In : 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS'14), Kyoto, Japan, pp.259-270 (2014)

8. Chen, K., Johnson, N., D'Silva, V., Dai, S., MacNamara, K., Magrino, T., Wu, E., Rinard, M., Song, D.: Contextual Policy Enforcement in Android Applications with Permission Event Graphs. In : 20th Annual Network and Distributed System Security Symposium, (NDSS'13), San Diego (2013)
9. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Traon, Y., Octeau, D., McDaniel, P.: FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In : 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'14), pp.259-269 (2014)
10. Jang, J.-w., Yun, J., Woo, J., Kim, H.: Andro-profiler: anti-malware system based on behavior profiling of mobile malware. In : WWW Companion '14 Proceedings of the companion publication of the 23rd international conference on World wide web companion, Seoul, Korea, pp.737-738 (2014)
11. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: Behavior- Based Malware Detection System for Android. In : ACM CCS workshop on Security and privacy in smartphones and mobile devices (SPSM'11), Chicago (2011)
12. Mulliner, C., Oberheide, J., Robertson, W., Kirda, E.: PatchDroid: Scalable Third-Party Security Patches for Android Devices. In : 29th Annual Computer Security Applications Conference (ACSAC'13), New Orleans, Louisiana, USA, pp.259-268 (2013)
13. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In : Network and Distributed System Security Symposium (2012)
14. Grace, M., Zhou, Y., Zhang, Q., Zou, S., Jiang, X.: RiskRanker: scalable and accurate zero-day android malware detection. In : 10th international conference on Mobile systems, applications, and services (MobiSys '12), pp.281-294 (2012)
15. Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T.: WHYPER: towards automating risk assessment of mobile applications. In : 22nd USENIX conference on Security (SEC'13), pp.527-542 (2013)
16. Enck, W., Gilbert, P., Chun, B.-G., Cox, L., Jung, J., McDaniel, P., Sheth, A.: TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones. Communications of the ACM 57(3) (March 2014)
17. Yan, L., Yin, H.: DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In : the 21st USENIX conference on Security symposium (Security'12), USENIX Association, Berkeley, CA, USA, pp.29-29 (2012)
18. Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., Wang, X., Zang, B.: Vetting undesirable behaviors in android apps with permission use analysis. In : 2013 ACM SIGSAC conference on Computer & communications security (CCS'13), pp.611-622 (2013)
19. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y.: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38(1), 161-190 (January 2011)
20. Dini, G., Martinelli, F., Saracino, A., Sgandurra, D.: MADAM: A Multi-level Anomaly Detector for Android Malware. In : Computer Network Security (Lecture Notes in Computer Science) 7531. Springer Berlin Heidelberg (2012) 240-253
21. Schmidt, A.-D., Peters, F., Lamour, F., Scheel, C., Çamtepe, S., Albayrak, S.: Monitoring Smartphones for Anomaly Detection. Mobile Networks and Applications 14(1), 92-106 (February 2009)
22. Xie, L., Zhang, X., Seifert, J.-P., Zhu, S.: pBMDS: a behavior-based malware detection system for cellphone devices. In : Third ACM conference on Wireless network security (WiSec '10), Hoboken, New Jersey, USA, pp.37-48 (2010)
23. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In : CCS '09 Proceedings of the 16th ACM conference on Computer and communications security, pp.235-245 (2009)

24. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically Rich Application-Centric Security in Android. In : ACSAC '09 Proceedings of the 2009 Annual Computer Security Applications Conference, Honolulu, HI, USA, pp.340-349 (2009)
25. Zhang, M., Duan, Y., Yin, H., Zhao, Z.: Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs. In : 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14), pp.1105-1116 (2014)
26. Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Poth, R., Nita-Rotaru, C., Molloy, I.: Using Probabilistic Generative Models for Ranking Risks of Android Apps. In : 2012 ACM Conference on Computer and Communications Security (CCS'12) (2012)
27. Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., Son, D.: Juxtapp: A Scalable System for Detecting Code Reuse Among Android Applications. In : 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'12) (2012)
28. Aafer, Y., Du, W., Yin, H.: DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. In : 9th International Conference on Security and Privacy in Communication Networks (SecureComm'13) (2013)
29. Arp, D., Spreitzenbarth, M., Huebner, M., Gascon, H., Rieck, K.: Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket. In : 21th Annual Network and Distributed System Security Symposium (NDSS) (February 2014)
30. smali. Available at: https://code.google.com/p/smali/
31. Android Emulator. Available at: http://developer.android.com/tools/help/emulator.html
32. Debian. Available at: http://www.debian.org/
33. MonkeyRunner Toolkit. Available at: http://developer.android.com/tools/help/monkeyrunner_concepts.html
34. Shridhar, D., Bartlett, E., Seagrave, R.: Information theoretic subset selection. Computers in Chemical Engineering 22, 613-626 (1998)
35. Shannon, C., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press (1949)
36. Patel, K., Buddhadev, B.: Predictive Rule Discovery for Network Intrusion Detection. In : Third International Symposium on Intelligent Informatics (ISI'14), Greater Noida, India, pp.287-298 (2014)
37. Gonzalez, H., Stakhanova, N., Ghorbani, A.: DroidKin: Lightweight Detection of Android Apps Similarity. In : International Conference on Security and Privacy in Communication Networks (SecureComm 2014) (2014)
38. Parkour, M.: ContagioDump. Available at: http://contagiodump.blogspot.in/