Classifying Anomalous Mobile Applications Based on Data Flows

Chia-Mei Chen
Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan
Email: cchen@mail.nsysu.edu.tw

Yu-Hsuan Tsai
Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan
Email: t0336470@gmail.com

Gu-Hsin Lai
Department of Information Management
Chinese Culture University
Taipei, Taiwan
Email: lgx4@ulive.pccu.edu.tw

Sheng-Tzong Cheng
Department of Computer Science and Info. Engineering
National Cheng Kung University
Tainan, Taiwan
Email: stcheng@mail.ncku.edu.tw

***Abstract*—Mobile security becomes more important as users increasingly rely on the portable network devices. The security consultant firms indicate that the amount of mobile malware increases every year at a fast speed. Therefore, fast detecting mobile malware becomes an important issue. By applying reverse engineering techniques, a source code extraction module produces data flow information from the mobile application executable. The proposed static analysis-based detection system analyzes the data flow of the target software and it identifies if a data flow might leak sensitive data. The experimental results show that the proposed detection system can identify mobile malware efficiently.**

Keywords- mobile security, malware detection, static analysis.

I. INTRODUCTION

Mobile users get used to downloading various mobile applications on the mobile devices for business as well as leisure purposes. Therefore, confidential information is stored in the mobile devices which become the new target for financial gain. Juniper Networks study [1] states that 92% of mobile malware targets the Android platform, as it has the highest market share. Tread Micro [3] reports that seventeen pieces of malware had already been downloaded seven hundred thousand times before they were removed and half of mobile malware involve unauthorized text message sending or network access. F-Security report [5] concludes that mobile malware are mostly profit oriented and security might be the primary concern for mobile users. The number of apps increases dramatically in the markets and an efficient mobile malware detection is demanded.

Commercial mobile malware detection solutions such as BullGuard Mobile Security and Lookout Mobile Security adopt signature-based approach [2] and the detection rate relies on the malware signature repository. For fast growing mobile malware, hackers have a chance to compromise mobile users before the signature is developed [14]. Hence, an alternative solution should be developed to detect unknown mobile malware.

In this research, the proposed detection system develops a feature selection method combining genetic algorithm and data flow analysis, where genetic algorithm reduces the number of features and data flow analysis shows the relationship between API calls and system commands. This research conducted a preliminary study analyzing collected mobile apps and malware and discovered that apps authors might obfuscate the codes by replacing variable names into meaningless strings but the API calls and system commands would not be altered. To steal privacy information, certain API calls and system commands would be invoked. Therefore, the proposed detection method considers the API calls and system commands as key attributes. Based on our preliminary study, the possible sequences of the API calls and system commands are huge. Therefore, genetic algorithm is applied to build efficient threat patterns of the API call and system command invocation. The proposed detection method can identify unknown malware which matches the malicious behaviors found.

The structure of the paper is organized as follows. The literature review is studied in Section II. Section III describes the proposed classification method, followed by performance evaluation in Section IV. The conclusion remarks are drawn in Section V.

II. RELATED WORK

Dynamic analysis and static analysis [21] are common approaches used for malware detection. Dynamic analysis consumes more resources and computation time, while static analysis requires source code or reverse engineering.

Bhaskar Pratim et al. [9] proposed an approach which analyzes the risk of an app based on permission. The approach is limited to the official Google Play market, but most malware resides in the third party markets. Francesco Di Cerbo et al. [11] applied Apriori algorithm to identify common subsets of permissions used by the benign apps.

As app writers may produce over-privileged mobile software [27][28], permission based approach might not be enough to identify mobile malware. Some malware even conducts malicious behaviors without permission [29]. Permission based mobile malware detection has drawbacks [20] and is not efficient.

William et al. [30] built an Android sandbox by modifying Android's source code. The sandbox traces the data flows of the sensitive data, such as IMEI or DeviceId, which appears in text messages or network connection. This method is designed for security researchers monitoring data flows in the mobile devices but not suitable for detecting mobile malware.

Shabtai et al. [14] proposed a detection system applying knowledge-based and temporal abstraction method to detect unknown malware. Temporal patterns of mobile devices are established from history events such as app installation and the number of text message sent out. A monitored event without user interaction is regarded as abuse. In the practical cases, users tend to press OK when using an app and hackers could apply social engineering tactics to circumvent such restriction.

Wu et al. [10] proposed a malware classification method which combines several types of features: permission and component information from Manifest file, information of intent, API calls and communication between components from source code. K-mean algorithm and expectation–maximization algorithm are applied to classify the mobile applications. Yerima et al. [7] proved that API calls and system calls are efficient for distinguishing malware and benign applications and Bayesian classifier is adopted to classify malwares and benign applications.

The above mentioned classification approaches do not provide the cause of malicious behaviors and might confuse users. The literature indicates that API calls and system calls are efficient and the invocation ordering is useful for defining malicious behaviors. Therefore, the proposed detection system develops an efficient feature selection method to identify efficient features and build the invocation sequences used by malware. With reduced feature sets, the proposed detection reduces the detection time without detection performance loss.

III. PROPOSED SYSTEM

The literature review and our preliminary study indicate that obfuscated software replacing variable names to meaningless strings makes static analysis based detection hard and each piece of software has unique invocation sequence. API call and system command invocation represents the behaviors of a piece of software. The distinct sequences of the invocations could increase large as the number of malware raises. Therefore, the proposed detection system develops a feature selection method which applies genetic algorithm to reduce the number of feature sets and build efficient threat patterns of API call and system command invocation.

The proposed system consists of three processes, reverse engineering, threat pattern building, and detecting processes as shown in Figure 1.

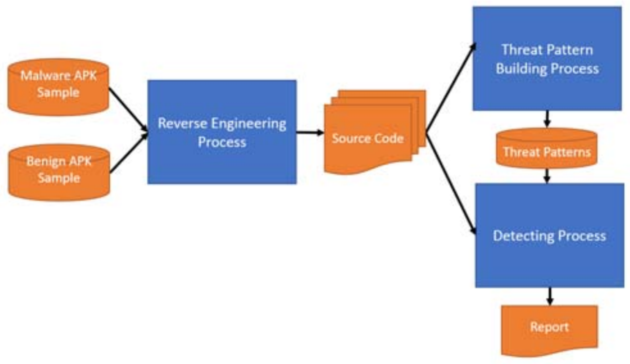


Figure 1. System architecture

Three tools, APKTool, dex2jar, and JAD, are applied for reversing app's APK files into source code. APKTool produces the .dex files from the apk files; dex2jar transforms the .dex files into a set of the .class files; and decompiler JAD converts the .class files into the .jad files which are the Java source code of the APK files. Source code provides valuable information. API calls and system commands can be retrieved.

Threat patterns are sequences of API calls and system command invocations. Some API calls and system commands are invoked by both malicious and benign apps, and are not distinguishable features for malware detection. Therefore, in the threat pattern building process, the feature selection module eliminates common calls and commands used by two types of the mobile applications.

Feature Set Reduction by Genetic Algorithm

Many API calls and system commands were found in the test dataset; therefore, the number of the possible combinations of the invocations is huge. The feature sets grow up as the number of invocation sequences increases. In this study, genetic algorithm is applied to select a suboptimal set of invocations which can distinguish mobile malware from the normal apps. The goal of the proposed classification system is to maximize the detection rate which is measured by true positive rate and precision in this study. Hence, the proposed fitness function is defined by the detection performance measurements mentioned above: true positive rate + precision.

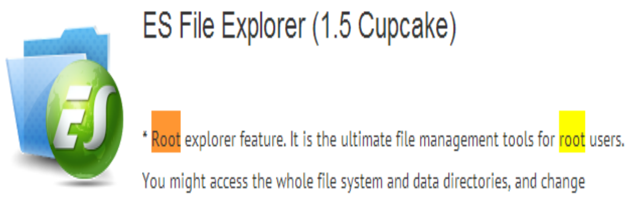
IV. SYSTEM EVALUATION

The mobile apps for evaluation were extracted from Android Malware Genome Project [26] and Google Play Market. This study assumes that the chance of a malicious and popular app which can survive in Google Play market for over three month is low.

Table I. DETECTION RESULTS.

Family	No of apps in the family	No. of detected malware	True positive
ADRD	22	22	100.00%
AnserverBot	186	187	99.47%
Asroot	7	8	87.50%
BaseBridge	115	122	94.26%
BeanBot	8	8	100.00%
Bgserv	9	9	100.00%
CoinPirate	1	1	100.00%
CruseWin	2	2	100.00%
DogWars	0	1	0.00%
DroidCoupon	0	1	0.00%
DroidDeluxe	1	1	100.00%
DroidDream	15	16	93.75%
DroidDreamLight	46	46	100.00%
DroidKungFu1	33	34	97.06%
DroidKungFu2	30	30	100.00%
DroidKungFu3	309	309	100.00%
DroidKungFu4	96	96	100.00%
DroidKungFuSapp	3	3	100.00%
DroidKungFuUpdate	1	1	100.00%
Endofday	1	1	100.00%
FakeNetflix	0	1	0.00%
FakePlayer	0	6	0.00%
GGTracker	1	1	100.00%
GPSSMSSpy	0	6	0.00%
GamblerSMS	1	1	100.00%
Geinimi	69	69	100.00%
GingerMaster	4	4	100.00%
GoldDream	47	47	100.00%
Gone60	0	9	0.00%
HippoSMS	2	4	50.00%
Jifake	0	1	0.00%
KMin	52	52	100.00%
LoveTrap	1	1	100.00%
NickyBot	1	1	100.00%
NickySpy	0	2	0.00%
Pjapps	57	57	100.00%
Plankton	11	11	100.00%
RogueLemon	2	2	100.00%
RogueSPPush	9	9	100.00%
SMSReplicator	1	1	100.00%
SndApps	10	10	100.00%
Spitmo	1	1	100.00%
Tapsnake	0	2	0.00%
Walkinwat	0	1	0.00%
YZHC	22	22	100.00%
Zitmo	0	1	0.00%
Zsone	12	12	100.00%
jSMSHider	16	16	100.00%
zHash	11	11	100.00%
Total	1215	1259	96.51%

The detection results are shown in Table I; the proposed system has the detection rate of 96.5%. The proposed detection method might have false negative on small size malware families, as the threat patterns used by them improve insignificantly on fitness function of the genetic algorithm. As for false positive, 119 benign samples out of 1,259 were classified as malicious. Some misclassified samples have root threat. For example, com.estrongs.android.pop.cupcak is one of the applications that being detected has root threat. As shown in Figure 2, the description of this application indicates that it requires root access. Most misclassified benign apps were detected as malwares because of data thief threat. For example, data synchronization app, com.gozap.labi.android copies information stored in the mobile device and sends to somewhere. Other misclassifications were caused by adware which sends out device ID for advertisement purpose [19]. Therefore, the results conclude that the proposed detection system can detect malware efficiently.



ES File Explorer (1.5 Cupcake)

* Root explorer feature. It is the ultimate file management tools for root users.

You might access the whole file system and data directories, and change

Figure 2. Misclassified benign app (snapshot of com.estrongs.android.pop.cupcak in Google Play) requires root access.

V. CONCLUSIONS

Mobile devices are widely used in our daily work and leisure time. The security surveys and reports demonstrate that hackers have shifted the attack target to mobile users and mobile malware increases each year. Signature based detection is not suitable for fast growing and changing mobile malware.

Static analysis is suitable for analyzing fast growing mobile malware. This study proposes a static analysis based detection method which identifies efficient feature sets from the API calls and system commands. Two phases of feature set reductions are developed and the experimental results show that the proposed detection using the feature selection method performs efficiently with the detection rate of 96.5%.

Further evaluation and investigation should be made to compare the proposed static analysis approach with signature based detection method and to analyze the process time required by the proposed system in the reverse engineering and model training phases.

Static analysis might have limitations. Malware with botnet capability which receives and executes attack commands from command and control server might not be detectable from static analysis.

The reverse engineering tools and techniques used in this study can be improved to extract better quality of source code. Some applications use NDK (Native Development Kit) which allows to develop functions in language C and to extend invocation via JNI (Java Native Interface). The C functions are compiled into share object (.so file) and hard to decompile back to the source code. The software which invokes malicious functions in C requires better detection and reverse engineering methods to identify the anomalous behaviors.

REFERENCES

[1] Juniper networks, “Juniper networks Mobile threat Center Third Annual Mobile threats report,” retrieved March 1, 2015 from http://www.juniper.net/us/en/local/pdf/additional-resources/3rd-jnpr-mobile-threats-report-exec-summary.pdf.

[2] Shuaifu Dai, “Behavior-Based Malware Detection on Mobile Phone” The 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010.

[3] TrendMicro, “Android Malware: How Worried Should You Be?” retrieved on March 1, 2015 from http://blog.trendmicro.com/trendlabs-security-intelligence/android-malware-how-worried-should-you-be/.

[4] McAfee, “McAfee Threats Report: Second Quarter 2013,” retrieved on March 1, 2015 from http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2013-summary.pdf.

[5] TrendMicro, “Android Malware: How Worried Should You Be?” retrieved on March 1, 2015 from http://blog.trendmicro.com/trendlabs-security-intelligence/android-malware-how-worried-should-you-be/

[6] F-Security, ”MOBILE THREAT REPORT Q4 2012,” http://www.f-secure.com/static/doc/labs_global/Research/Mobile%20Threat%20Report%20Q4%202012.pdf.

[7] Yerima, S. Y., Sezer, S., McWilliams, G., and Muttik, I., “A New Android Malware Detection Approach Using Bayesian Classification,” the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2013.

[8] Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., and Olson, A. J., “Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function,” Journal of Computational Chemistry, 1998.

[9] Bridge, D., “Genetic Algorithms,” retrieved on March 1, 2015 from http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout12.pdf.

[10] Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., and Molloy, I., “Android permissions: a perspective combining risks and benefits,” The 17th ACM symposium on Access Control Models and Technologies, 2012.

[11] Wu, D.J., Mao, C. H., Wei, T. E., Lee, H. M. and Wu, K. P., “DroidMat: Android Malware Detection through Manifest and API Calls Tracing,” The 7th Asia Joint Conference on Information Security, 2012.

[12] Di Cerbo, F., Girardello, A., Michahelles, F., and Voronkova, S., “Detection of malicious applications on android OS,” Computational Forensics, 2011.

[13] Enck, W., Ongtang, M., & McDaniel, P., “On lightweight mobile phone application certification,” The 16th ACM conference on Computer and communications security, 2009.

[14] Chiang, W. C., “Behavior Analysis of Mobile Malware Based on Information Leakage”, master thesis of National Sun Yat-sen University, 2013

[15] Shabtai, A., Kanonov, U., and Elovici, Y., “Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method,” in Proc. Journal of Systems and Software, vol. 83, no. 8, 2010, pp. 1524-1537

[16] Cover, T. M., and Thomas, J. A., “Entropy, relative entropy and mutual information,” Elements of Information Theory, 1991.

[17] Kouznetsov, P., “JAD Java Decompiler,” retrieved on March 1 2015 from http://www.varaneckas.com/jad/.

[18] Lin, J. M., ”Detecting Mobile Application Malicious Behavior Based on Taint Propagation”, master thesis of National Sun Yat-sen University, 2013

[19] Aafer, Y., Du, W., and Yin, H., “DroidAPIMiner: Mining API-level features for robust malware detection in android,” Security and Privacy in Communication Networks, 2013.

[20] Blasing, T., Batyuk, L., Schmidt, A. D., Camtepe, S. A., and Albayrak, S., “An android application sandbox system for suspicious software detection,” The Fifth international IEEE conference on Malicious and Unwanted Software (MALWARE), 2010.

[21] McAfee Lab , 2012, “FakeInstaller’ Leads the Attack on Android Phones,” retrieved on March 1, 2015 from https://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones

[22] Richardson, L., retrieved on March 1, 2015 from “BeautifulSoup,” http://www.crummy.com/software/BeautifulSoup/.

[23] Neumann, M., “Mechanize,” http://mechanize.rubyforge.org/.

[24] Zhou, Y. and Jiang, X., Android Malware Genome Project, retrieved March 1, 2015 from http://www.malgenomeproject.org/.

[25] Adrienne Porter Felt, Kate Greenwood, and David Wagner, “The effectiveness of application permissions,” The Second USENIX Conference on Web Application Development, 2011.

[26] Wei, X., Gomez, L., Neamtiu, I., and Faloutsos, M., “Permission evolution in the android ecosystem,” The 28th ACM Annual Computer Security Applications Conference. ACM, 2012.

[27] Grace, M. C., Zhou, Y., Wang, Z., and Jiang, X, “Systematic Detection of Capability Leaks in Stock Android Smartphones,” The Annual Network & Distributed System Security Symposium (NDSS), 2012.

[28] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., and Sheth, A. N., “TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones,” The 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI’10), 2010.