Final Project Proposal

Jia Yaoqi*[1], Fan Qi†[1], Michael Atmadja‡[1], and Sesha Sendhil§[1]

[1]School of Computing, National University of Singapore

February 8, 2013

1 Introduction

Smartphones are becoming worldwide and replacing desktop as the main personal computing platforms. A recent report from Gartner[7] shows that worldwide sales of mobile phones to end users reached almost 428 million units in the third quarter of 2012. Unfortunately, the increasing adoption of smartphones comes with the growing prevalence of mobile malware. As the most popular mobile platform, Android faces 46% share of malware among all mobile platforms[10]. There is a pressing need to effectively mitigate or defend against the malware.

2 Related work

Smartphone security and privacy has recently become a major concern. TaintDroid[4] and PiOS[3] are two representative systems that demonstrate potential privacy threats from third party apps in both Android and iOS platforms. Comdroid[2] analyzes the vulnerability in inter-app communication in Android apps. Stowaway[6] examines and finds over-privileged apps. Kirin [5]blocks the installation of potential unsafe apps if they contain dangerous permission combination. Apex[9], MockDroid[1], TISSA[12] and AppFence[8] revise the current Android framework to provide fine-grained controls of resources for third-party apps.

*jiayaoqi@comp.nus.edu.sg
†fan.qi@nus.edu.sg
‡michael.a@nus.edu.sg
§seshasendhil@gmail.com

3 Methodology

The main purpose of our project is to develop a pratical malware detection tool to detect suspicious apps on Android. Instead of using traditional ways to detect malware by signatures, we use suspicious behaviors to define malware. Suspicious behaviors include dynamically fetching and executing code from a remote untrusted websites, loading native code in an unusual ways and other behaviors.

3.1 platforms and tools

Ubuntu 10.04
Eclipse + ADT plug-in
Android emulator
Galaxy Nexus
apktool(for reverse engineering 3rd party, closed, binary Android apps)
Dare(retarget Android applications in .dex or .apk format to traditional .class files)
Other reverse engineering tools

3.2 Analysis of Malware

In first step, we try to read some related papers and analyze different kinds of malware samples. Then figure out how malware install, activate and their payload. Then try to find out the inner behaviors behind different actions. We will change Android API, and add log function on each class and method . The aim of this action is to log and audit each running app's calling methods and classes sequences for next step's analysis.

3.3 Heuristics-based detecting scheme

In second step, we will use machine learning and other methods to analyze the API calls sequences to find the regular pattern of malware. Then utilize the regular pattern of malware to detect suspicious apps and distinguish whether the app is malicious or not. Besides, since we log all the API calls of the running app, we can pop up notifications when the app has suspicious behaviors, such as send messages to subscribe premium services and upload sensitive data like contact to remote servers.

3.4 reverse engineer app protocols

Another novel idea is that we want to reverse engineer app protocols to figure out how apps communicate with their servers(if they have one), and try to hack the server by using local end's private information contained by protocol, Like Rui Wang's work[11] on SSO web services. For example, a popular app may permit user to login with Google account. By this way, the app must implement Google account API on their own source code. However, Google does not check how their API are utilized on other apps and the app developers do not check whether it is secure when they implement the open API. Then the protocol among user, app and Google account may be not as secure as ssl. So we may reverse engineer the protocol and analyze the vulnerability to do spoof attack.

4 Time plan

In first month, read related papers and get familiar with reverse engineer tools, malware and Android APIs.
In second month, modify Android APIs to add log and audit functions, compile Android source code. Then run malware to record their API calls sequences. Run benign apps and do the same things. At last, use SVM or other tools to train the records, then classify the suspicious apps with the training data.
In the third month, we utilize the lookup and other third-party malware detection tools to detect the samples(including both malware and benign apps) and compare the results with our tool by false positive, false negative, overhead on cost time and battery. Then tabulate tables and draw graphs to demonstrate discrepancy explicitly.

References

[1] A.R. Beresford, A. Rice, N. Skehin, and R. Sohan, *Mockdroid: trading privacy for application functionality on smartphones*, Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ACM, 2011, pp. 49–54.

[2] E. Chin, A.P. Felt, K. Greenwood, and D. Wagner, *Analyzing inter-application communication in android*, Proceedings of the 9th international conference on Mobile systems, applications, and services, ACM, 2011, pp. 239–252.

[3] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, *Pios: Detecting privacy leaks in ios applications*, Proceedings of the Network and Distributed System Security Symposium, 2011.

[4] W. Enck, P. Gilbert, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, *Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones*, Proceedings of the 9th USENIX conference on Operating systems design and implementation, 2010, pp. 1–6.

[5] W. Enck, M. Ongtang, and P. McDaniel, *Mitigating android software misuse before it happens*, (2008).

[6] A.P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, *Android permissions demystified*, Proceedings of the 18th ACM conference on Computer and communications security, ACM, 2011, pp. 627–638.

[7] Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent, /url-http://www.gartner.com/newsroom/id/2237315.

[8] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, *These aren't the droids you're looking for: retrofitting android to protect data from imperious applications*, Proceedings of the 18th ACM conference on Computer and communications security, ACM, 2011, pp. 639–652.

[9] M. Nauman, S. Khan, and X. Zhang, *Apex: extending android permission model and enforcement with user-defined runtime constraints*, Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ACM, 2010, pp. 328–332.

[10] Number of the Week: at Least 34 percentage of Android Malware Is Stealing Your Data, /url-http://www.kaspersky.com/about/news/virus/2011.

[11] R. Wang, S. Chen, and X.F. Wang, *Signing me onto your accounts through facebook and google: a traffic-guided security study of commercially deployed single-sign-on web services*, Security and Privacy (SP), 2012 IEEE Symposium on, IEEE, 2012, pp. 365–379.

[12] Y. Zhou, X. Zhang, X. Jiang, and V. Freeh, *Taming information-stealing smartphone applications (on android)*, Trust and Trustworthy Computing (2011), 93–107.