Platform Neutral Sandbox for Analyzing Malware and Resource Hogger Apps

Parvez Faruki[1](✉), Vijay Kumar[1], Ammar B.[1], M.S. Gaur[1], Vijay Laxmi[1], and Mauro Conti[2]

[1] Department of Computer Engineering, Malaviya National Institute of Technology, Jaipur, India
{parvez,vlaxmi}@mnit.ac.in, {vijay.ganmoor,bharmal.ammar,gaurms}@gmail.com

[2] University of Padua-Department of Mathematics, Padua, Italy
conti@math.unipd.it

Abstract. In this paper, we propose an automated, scalable, and dynamic analysis framework incorporating static *anti anti-analysis* techniques to detect the analysis environment aware Android malware and Resource Hogger apps. The proposed framework can automatically trigger malicious execution by sending simulated User-Interface (UI) events and `Intent` broadcasts. The Proposed approach is *scalable* and platform invarient for different Android OS versions.

Keywords: Dynamic Analysis · Environment Reactive Behavior · Resource Hogger Apps

1 Introduction

Smartphone stores personal information, thus privacy and security of device is the prime concern. Android devices control 2/3 market presence among the total smartphone [4]. Android platform secures apps by: (1) sandboxing app execution (2) Permissions based access model [2]. Anti-malware apps protects devices, but cannot detect unseen variants or zero-day malware [11].

In this paper we propose a scalable, dynamic analysis framework to analyze and detect Android malware, Resource Hoggers and data leaking apps. Privacy risk apps may leak user information such as smartphone identification number (IMEI), subscriber identification (IMSI) without user knowledge. We execute Android apps in an emulated environment enriched with static anti anti-analysis capability to detect environment reactive malware. Proposed Sandbox monitors file operations, downloads, suspicious payload installation. Proposed approach also monitors aggressive app behavior such as contacting URLs and exhausting network bandwidth.

The paper is organized as follows. Section 2 defines proposed methodology, its salient features and anti anti-analysis environment. Section 3 covers experimental setup, analysis and comparison with prominent existing frameworks. Finally, Sect. 4 concludes this paper with pointers to future work.

2 Proposed Methodology

The essence of the proposed dynamic analysis sandbox is its multiple analysis methods to detect malicious apps as depicted in Fig. 1. When an app is submitted to the Sandbox, clean isolated environment is initialized with a refreshed Android emulator with clean OS snapshot for a quick start. Android Virtual Device (AVD) manager [1] allows creation, saving and snapshot restore and load the emulator. The Sandbox starts the emulator(s) with *save-to-snapshot* functionality to resemble it as a real device by adding wallpaper, messages, contacts and setting custom device settings. Each time an app is submitted for analysis, clean emulator snapshot is loaded.

As shown in Fig. 1, Framework core controls all the components for essential feature collection, facilitating the AVD loading, and generating analysis reports. Dalvik Dynamik Instrumentation (DDI) hooking libraries are used to hook various methods that are helpful in behavior monitoring. Analysis module results are summarized to predict malicious, resource hogger, potential risk or a benign app. Proposed sandbox employs DDI to identify resource hoggers and privacy risk apps.

Features of Proposed Sandbox

The analysis environment sets up static anti *anti-analysis* features to modify static emulator properties to resemble it as real device. Proposed Sandbox is scalable as we employ a transparent functionality without modifying the Android platform. Resource Hogger App detection is based on anomalous consumption of CPU, memory or network resource consumption in comparison to benign apps.

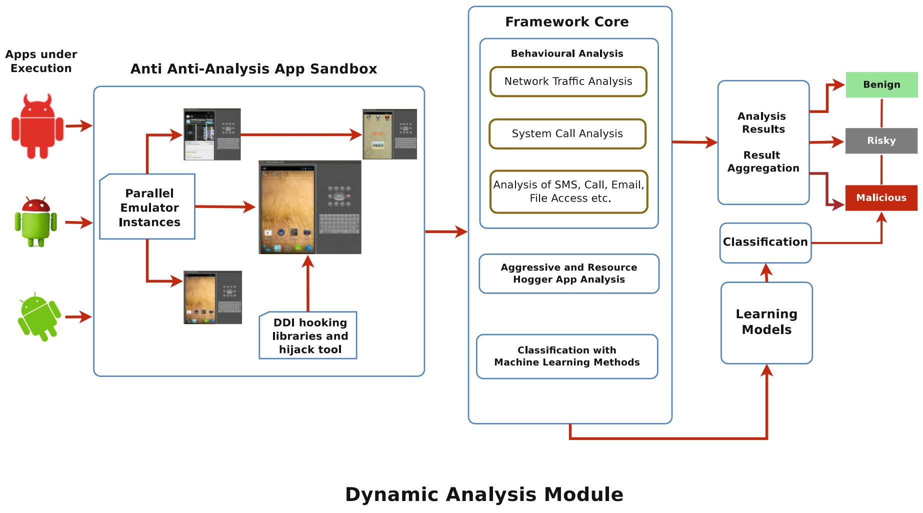
Apps under Execution
Anti Anti-Analysis App Sandbox
Parallel Emulator Instances
DDI hooking libraries and hijack tool
Framework Core
Behavioural Analysis
Network Traffic Analysis
System Call Analysis
Analysis of SMS, Call, Email, File Access etc.
Aggressive and Resource Hogger App Analysis
Classification with Machine Learning Methods
Analysis Results
Result Aggregation
Classification
Learning Models
Benign
Risky
Malicious
Dynamic Analysis Module

Fig. 1. Proposed Dynamic Analysis Approach

Proposed approach finds a strong link between malware and its resource usage pattern. App tagging is performed based on its behavior.

Anti anti-analysis Features

Targeted malware families *Bgserv* and *AnserverBot* use static *anti-analysis* techniques to avoid analysis environment detection and bypass the default analysis configuration parameters. Proposed framework generates static *anti anti-analysis environment* and resembles the emulated device as real. `IMEI`, `IMSI`, serial number, phone number defaults are modified to resemble real device. Geolocation properties, system time, e-mail account configuration, wallpaper, images and audio/video files are added to the standard emulator. These static changes to the analysis environment correspond to real devices, hence we have been able to uncover quite a few anti-analysis malware.

Revealing App Behavior with Triggers

Proposed behavioral analysis framework lures the apps and provides them with required events to force reveal the malicious functionality. Triggers such as `Intents`, `SMS` sent/received, app activation time etc. are important triggers for malware activation. We find all the `Intents` needed by the app and generate them using Android Debug Bridge (ADB) to initiate corresponding component (i.e., activity, service or broadcast receiver). We generate few implicit `Intent`(s) such as `SMS_SENT` or `NEW_OUTGOING_CALL` and explicitly generate other `Intents`. In case of time triggered actions, AVD system time is set to some future time with fixed interval(s). Automated user inputs are generated with `monkey` [1].

Behavioral Analysis. After recording the app actions, we analyze them with behavioral analysis with `logcat` results to detect any installations, new process spawns and `SMS` sent. We scan the traffic `(.pcap)` files to analyze malicious URLs' or sensitive information leakage. Analysis of system calls relates file and network related activities. Proposed analysis reports few system calls (bind and connect) prominently visible among malware apps, hence an app with such calls is considered risky. Proposed Sandbox marks actions like sending `SMS`, e-mail(s) without user consent as covert misuse of existing facilities not seen among normal apps. Sending private user data such as call logs, contacts, existing `SMS` and e-mails, encrypting sensitive user data (contacts, SMS), GPS co-ordinates activities not visible in normal apps, hence considered grave potential risk.

Dalvik Dynamic Instrumentation (DDI). DDI [6] hooks itself to classes and methods of Dalvik Virtual Machine (DVM). We use DDI to observe various runtime strings to detect encrypted malware. Framework Instruments hooks for `SMSManager` class to keep track of messages sent by an app and `Intent` class to monitor phone calls and e-mails.

Resource Hoggers or Aggressive Malware Analysis. An app is categorized aggressive when resource usage pattern is anomalous compared to benign usage. Monitoring memory consumption, network usage (URLs, bandwidth consumed), CPU utilization, battery utilization is useful to detect behavioral anomaly. We analyzed the comparative resource usage among a pool of categorized benign and malware apps and generate experimental threshold to detect anomalous resource usage.

3 Experimental Set-Up

Proposed model sets up multiple emulator(s) in parallel. Submitted apps are concurrently divided among free emulators. The Sandbox loads emulator(s) with a clean snapshot for quick start. ADB interacts with the emulator(s) for data collection. Proposed Sandbox utilizes recording tools such as logcat, tcpdump, monkey, strace and dumpsys [1] and prefers a scalable environment without any modification to the existing OS.

Aggregated Analysis

Proposed Sandbox has a rich set of multiple analysis techniques capable of predicting the app behavior. To minimize false positives, malware prediction is set by the behavioral detection module. Proposed detection model marks an app malware if malicious behaviors are discovered and are justifiable as purely malicious. If all three modules cannot find anomaly within monitored app, the Sandbox declares it as a benign.

Comparison with Existing Works

Droidbox and Taintdroid form base of other existing dynamic frameworks such as Andrubis, Apps Playground and SmartDroid. Proposed analysis employs Sandbox as a platform-neutral and scalable analysis environment. A similar approach

Table 1. Comparison of proposed framework with existing works

Property	**AASandbox [3]**	**Andromaly [8]**	**Apps Playground [7]**	**Droidbox [10]**	**Andrubis [5]**	**Proposed Approach**
Transparent & Scalable	✔	✔				✔
Resource Consumption		✔				✔
API Hooking			✔	✔	✔	✔
Logcat Analysis						✔
System-call Analysis	✔					✔
Risk Prediction	✔	✔			✔	✔
Anti *Anti-Analysis*			✔	✔	✔	✔
Identifying Data Leakage			✔	✔	✔	✔
Identifying SMS/Call Misuse			✔			✔
Network Traffic Analysis				✔	✔	✔
File Operations Monitoring				✔	✔	✔

to target quite different target has also been adopted in [9]. Table 1 compares proposed approach with some techniques.

4 Conclusion and Future Work

In this paper, we have proposed a platform neutral, scalable dynamic analysis framework that uncovers targeted and advanced Android malware, Resource Hoggers and risky apps equipped with anti anti-analysis capabilities. To the best of our knowledge, proposed framework for the first time integrates novel features such as platform neutral, scalability and DDI monitoring. Preliminary results suggest a corelation between Android malware and heavy resource utilization. In future, we aim to integrate dynamic anti anti-analysis techniques and perform large scale app analysis on the as a web based malware app detection framework.

References

1. Android tools: Adb, emulator, avd manager, android, mksdcard, monkey, logcat. http://developer.android.com/tools/help
2. Android Security Overview. http://source.android.com/devices/tech/security (Online last accesed on 24 April 2014)
3. Blasing, T., Batyuk, L., Schmidt, A.-D., Camtepe, S.A., Albayrak, S.: An android application sandbox system for suspicious software detection. In: 5th International Conference on Malicious and Unwanted Software (MALWARE), 2010, pp. 55–62. IEEE (2010)
4. G. Inc., Android Smartphone Sales Report (2013). http://www.gartner.com/newsroom/id/2665715 (online last accessed 17 March 2014)
5. Lindorfer, M.: Andrubis: a tool for analyzing unknown android applications. http://blog.iseclab.org/2012/06/04/andrubis-a-tool-for-analyzing-unknown-android-applications-2/
6. Mulliner, C.: Dalvik dynamic instrumentation, October 2013. http://www.mulliner.org/android/feed/mulliner_dbi_hitb_kul2013.pdf
7. Rastogi, V., Chen, Y., Enck, W.: Appsplayground: automatic security analysis of smartphone applications. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy, CODASPY 2013, pp. 209–220. ACM, New York (2013)
8. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y.: "Andromaly": a behavioral malware detection framework for android devices. J. Intell. Inf. Syst. **38**(1), 161–190 (2012)
9. Suarez-Tangil, G., Conti, M., Tapiador, J.E., Peris-Lopez, P.: Detecting targeted smartphone malware with behavior-triggering stochastic models. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 183–201. Springer, Heidelberg (2014)
10. Yan, L.K., Yin, H.: Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security 2012, pp. 29–29. USENIX Association, Berkeley (2012)
11. Zheng, M., Lee, P.P.C., Lui, J.C.S.: ADAM: an automatic and extensible platform to stress test android anti-virus systems. In: Flegel, U., Markatos, E., Robertson, W. (eds.) DIMVA 2012. LNCS, vol. 7591, pp. 82–101. Springer, Heidelberg (2013)