A Study on Abnormal Behaviour in Mobile Application

Naqliyah Bt Zainuddin*, Mohd Faizal Bin Abdollah, Robiah Bt Yusof, Shahrin Bin Sahib

Faculty of Information and Communication Technology, University Technical Malaysia Malacca, Karung Berkunci No. 1752 Pejabat Pos Durian Tunggal, Melaka, Malaysia
Email: *naqliyah@cybersecurity.my

Received 31 October 2014; revised 2 December 2014; accepted 26 December 2014

Abstract

Abnormal application behavior in mobile can produce a number of undesirable effects. An incorrect or insufficient implementation of application lifecycle, memory related issues and malicious application might cause an unexpected behavior of the application such as bad usability, not responding, crashed and even data loss. Current analysis and detection of abnormal applications behavior are still not comprehensive enough where behavior under user visible failure category such as crash, "stopped unexpectedly" and "not responding" received less attention by researchers. Furthermore, framework of analysis technique has not been developed by researcher to investigate the abnormal behavior in mobile application. Thus, in this paper we will study, analyze and classify the possible issues in causing abnormal application behavior and the existing techniques in identifying abnormal application behavior.

Keywords

Abnormal Behavior, Application, Android, Analysis Techniques

Subject Areas: **Applications of Communication Systems, Mobile and Portable Communications Systems**

1. Introduction

In today's world, mobile applications are becoming increasingly important in all aspects of our lives. No longer are phones reserved just for making calls, they now do more than the PC's of a few years ago. The open source Android operating system is a great example of the future of mobile applications. The rapid growth of smartphones has lead to a renaissance in mobile application services. Android and iOS, currently the most popular smartphone platforms, each offer their own public marketplace.

*Corresponding author.

How to cite this paper: Zainuddin, N.B., Abdollah, M.F.B., Yusof, R.B. and Sahib, S.B. (2014) A Study on Abnormal Behaviour in Mobile Application. *Open Access Library Journal*, **1**: e1229. http://dx.doi.org/10.4236/oalib.1101229

Detection of malwares, resources issues and others factors causing unexpected or abnormal behavior in mobile application has been the main focus by researchers in mobile security. As stated by [1], the major focus on Android security research is analyzing application for malicious behaviour.

[2] in his research also highlighted about "AppBrain", a website which provide a mechanism called "low quality application detection". It acts as a filter by automatically perform detection of applications which are unlikely to be useful or "low quality applications". This resulted Google to remove these applications from the market roughly once a quarter, in which case the total number of available Android applications goes down. The removed applications are almost always classified by as "low quality applications".

Even though there is no specific definition on "low quality application", inevitably, low quality applications can produce a number of undesirable effects and caused abnormal application behavior in mobile.

Thus, this research will study, analyze and classify the possible issues in causing abnormal application behavior and the existing techniques in identifying abnormal application behavior.

2. Abnormal Behavior in Mobile Application

The word "abnormal" means deviating from what is normal or usual, typically in a way that is undesirable, unexpected or worrying. [3] highlighted that, anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior. [4] added that anomaly detection techniques commonly used the following theories such as probability and statistics, artificial neural networks, genetic algorithms, fuzzy recognition and artificial immune method.

According to [5], crash event is identified as "abnormal behavior" because it is an unpredictable event that occurs when the system is in an arbitrary state and can produce a number of undesirable effects. A crash is defined as a fatal condition that occurs when a piece of software stops performing the activities it has been designed for.

Furthermore, [6] highlighted that an unexpected behavior or crashes in software systems can be similarly avoided by monitoring the behavior of constituent methods and modules, if we know which methods or modules are likely to cause a software crash beforehand. Another definition by [7] categorized crash as a "user visible failure", when a system alert displaying the message "Force Close" (in Android 2.2) or "Application has stopped unexpectedly" (in Android 4.0). These failure messages manifest in the log files as a log entry stating "FATAL EXCEPTION: main" and are essentially effects of uncaught exceptions thrown by the Android runtime.

Malicious software will also resulted in unexpected behavior by attempting to leak personal information, getting root privilege and abuse functions of the mobile [8]. [9] had stressed that even if applications have acquired explicit user consents, users may be unaware that the applications may execute malicious behaviors. Besides, [10] also highlighted other standard malicious attacks for PCs, like worms and Trojans are also becoming applicable to the mobile platforms. Malicious software such as Geimini and Droid Dream will result in unexpected behavior by attempting to leak personal information, getting root privilege and abuse functions of the smartphone as reported by [8]. [11] also had reported that the behavior of malicious applications could vary from annoying messages to very unrecoverable damages.

Definitely, a compromised smartphone can inflict severe damages and caused unexpected behavior in Android application. Memory leaks are highlighted by [12] as one of the major issues seen on the performance side of the mobile application which causing a sluggish behavior. [13] and [14] also emphasized that the memory leak phenomenon will affect the memory usage, affects the application to switch efficiency and cause the increase of memory usage and diminish overall system performance.

Despite the capability of Android to handle memory allocation using garbage collection automatically, [15] in his research identified that many applications currently suffer from memory leak vulnerabilities and causing applications to crash due to out of memory error while running.

Based on literature review, this study has managed to classify the abnormal behavior in mobile application as depicted in **Figure 1**.

Figure 1 summarized the general classification of abnormal behavior in mobile application. For "user visible failure", the application behavior under this category are crash, "application not responding" and "application has stopped unexpectedly". This type of behavior is sharing similar characteristics where it is an unexpected type of behavior and visible to users. For "user invisible failure", data leakage and unauthorized access are examples of an unexpected behaviors and invisible to users. This list of classification is not an exhaustive list and it may include other type of application activity with same behavior.

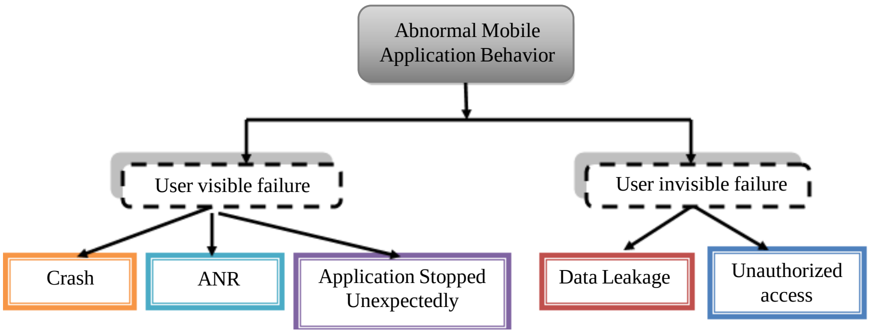

Figure 1. General classification of abnormal behavior in mobile application.

Above researchers highlighted the possible reasons on unexpected or abnormal behavior in an android application. Despite the outbreak of research activity in this area, [16] has highlighted that there is no framework yet that focuses on analysis and profiling the behavior of an Android application. Definitely, abnormal behavior in mobile application can produce a number of undesirable effects which might cause an unexpected behavior such as bad usability, not responding, crashed and even data loss. Majority of works done are focusing on detecting of malicious behavior due to malicious software whereas less work done so far in identifying abnormal application behavior which causing application to crash, “stopped unexpectedly” and “not responding”. In the next section, this study will explore related work done on the behavior related detection technique and analysis on mobile application.

3. Related Works in Detecting Abnormal Behavior in Mobile Applications

“CrowDroid” is a framework introduced by [17]. The framework is using dynamic analysis on system call (Strace) which enable the distinguishing between applications that having the same name and version but behave differently. The focus of the framework is to detect anomalously application in form of Trojan horses. CrowDroid used Strace to output the behavior patterns such as system calls of installed applications on users’ devices. This information is sent to a remote server where the system calls are clustered using a K-means algorithm into benign and malicious categories. CrowDroid concluded that open (), read (), access (), chmod () and chown () are the most used system calls by malware. Moreover, [18] introduced “Andromaly” another behavioral malware detection framework for android devices. Andromaly is a lightweight malware detection system using Machine Learning classification techniques to classify collected observations (system performance, user activity, memory, CPU consumption, battery exhaustion etc.) as either normal or abnormal.

Another work is by [19] proposed “AASandbox” (Android Application Sandbox). AASandbox is using static and dynamic approach to automatically detect suspicious application. For static approach, AASandbox scans the software for malicious patterns without installing it. While for dynamic approach, the analysis on the application is conducted in fully isolated environment which intervenes and logs low-level interactions. [20] had introduced a comprehensive software inspection framework. The framework allows identification of software reliability flaws and to trigger malware without require source-code. The framework is using dynamic approach by collecting run-time behavior analysis and also the I/O system calls generated by the applications.

[21] had introduced “ModelZ” for monitoring, detection, and analysis of energy-greedy anomalies in mobile handsets. Using light weight approach, ModelZ will monitor, detect and analyze new or unknown threats and energy-greedy anomalies on small mobile devices, with high accuracy and efficiency. [22] introduced “DroidBox” a dynamic analysis tool to classify Android applications by monitoring API calls of interest invoked by an application. The analysis includes generating two graphs (behavior graphs and treemap graphs) for sample in order to provide the basis in identifying benign or malicious categories.

[23] also had used system call, logs and timestamp information in his research to detect the “misbehaving” applications, alert the users, and log the evidence of malicious activities with. From the discussion on analysis technique in detecting malicious application, Strace is identified as a common tool in Android research and it has been used in works on malware detection by most of the researchers. Strace used the view of Linux-kernel such as network traffic, system calls, and file system logs to detect anomalies in the Android system. Furthermore, [17] also emphasized that monitoring systems calls (Strace) is one of the most accurate techniques to de-

termine the behavior of an Android application since they provide detailed low level information. In the next section, we will discuss on other analysis technique used by researchers in analyzing other type of abnormal behavior due to resources leaks and application life-cycle.

The detection of resources problems in mobile application has been studied by [24], [25] and [14]. [24] introduced an approach using static analysis tools called Relda, which can automatically analyze an application's resource operations and locate resource leaks. The method is based on a modified Function Call Graph, which handles the features of event-driven mobile programming by analyzing the callbacks defined in Android framework.

[25] proposed a novel and comprehensive approach for systematic testing for resource leaks in Android application. The approach is based on a GUI model, but is focused specifically on coverage criteria aimed at resource leak defects. These criteria are based on neutral cycles: sequences of GUI events that should have a "neutral" effect and should not lead to increases in resource usage.

The work on memory leakage detection is by [14] using a PCB hooking technique. The technique is using dynamic analysis by gathering memory execution information (*i.e.*; process ID, priority, shared library list, specific process-resource list) in run-time to detect memory leakage. In the experiment, Memory Analysis Tool (MAT) was used as a comparison with their invented tool.

The only work on monitoring software crashes is by [6] who presented a framework which monitors and reproduces software crashes. This approach involves learning patterns from features of methods that previously crashed to classify new methods as crash-prone or crash-resistant. Investigations had shown that 30% of crashed methods in ECLIPSE and 44% from ASPECTJ threw exceptions. The remaining 70% of crashed methods are not throwable and it is less common to see developers throw runtime exceptions in their programs.

Futhermore, [26] presented a tool called "AndroLIFT" which helps the developer to monitor the life cycle, assists in implementing it and testing life cycle-related properties. AndroLIFT is written as an extension to the ADT, the common way of developing Android applications with the Eclipse IDE. The life cycle view of this tool allows the developer to observe and analyze the life cycle of the Android application. Besides, it allow developer to easily learn about the behavior of the application life cycle to certain triggers, like an incoming call, and with which callback methods one can react appropriately. The summary of analysis techniques used in the detecting malicious and abnormality in mobile application is depicted in **Table 1**.

All in all, the aforementioned frameworks and systems as stated in **Table 1** proved valuable in protecting mobile devices in general. Most of the works are focusing on malware detection in mobile application using both dynamic and static analysis techniques. Detection technique on malicious software received a lot of attention by researchers. However, there is a gap in identifying the abnormal behavior which may lead to behavior of crash, "stopped unexpectedly" and "not responding".

From the discussion on analysis technique in detecting malicious application, *Strace* is identified as a common tool in Android research and it has been used in works on malware detection by most of the researchers. *Strace* used the view of Linux-kernel such as network traffic, system calls, and file system logs to detect anomalies in the Android system. Furthermore, [17] also emphasized that monitoring systems calls (*Strace*) is one of the most accurate techniques to determine the behavior of an Android application since they provide detailed low level information.

Moreover, [26] has highlighted that logcat is identified as the main logging mechanism in mobile application. Logcat allows us to capture the system debug output and log messages from the application. Wei [16] used a combination of the *logcat* and *getevent* tools of ADB to gather the data of the user layer for multi-layer profiling of Android application.

A specific tool for memory analysis is Memory Analyzer Tool (MAT). The *MAT* tooling is a set of plug-ins which visualizes the references to objects based on Java *heap dumps* and provides tools to identify potential memory leaks in Android applications. *The MAT* detects leakage by analyzing heap memory of one application. MAT analyzes heap memory situation when extracting log, and shows information which turns into a cause of memory leakage defect [14].

4. Proposed Framework and Summary

The framework of analysis techniques for abnormal application behavior is proposed as a way to identify the reasons of abnormal activity in mobile application. In this study, analysis techniques are described and applied

Table 1. Analysis techniques in detecting abnormal behavior in mobile application.

Works related	Category	Criteria of detecting abnormal behavior
Model Z Kim (2011)	Energy-greedy anomalies	Monitor and record usage of software and hardware resources
Crow Droid Burguera *et al.* (2011)	Malicious software	Using *Strace* to output the behavior patterns such as system calls
Andromaly Shabtai *et al.* (2011)	Malicious software	Using Machine Learning Classification to classify collected observation information
AAS and box Bl *et al*. (2010)	Malicious software	Intervenes and logs low-level interaction of an apps
Karami *et al*. (2013)	Malicious software	Collecting run-time behavior analysis and also I/O system calls generated by an apps
Isohara *et al.* (2011b)	Malicious software	Using log collector to record activity on kernel layer
Guo *et al.* (2013)	Resource leaks	Using Function Call Graph
Yan (2013)	Resource leaks	Using GUI model to detect resource leaks defect
Park *et al.* (2012)	Memory leakage	Using PCB hooking technique to gather memory execution information
Kim *et al.* (2010)	Crash method	Learning patterns from features of the method that previously crashed
AndroLIFT Franke *et al.* (2012)	Monitor apps life-cycle	Using an extension to ADT
DroidBox Alazab *et al.* (2012)	Malicious software	Monitoring API calls of interest invoked by an apps
Thing *et al.* (2011)	Malicious software	Using *strace* to log the system call, logs and timestamps information invoked by an apps
Wei (2013)	Profiling of android application	Measure and profile the apps at four layers

to Android applications to identify causes of abnormal behavior. The proposed framework of analysis techniques will utilize a combination of Linux trace (Strace) and Android debug facilities techniques (logcat and MAT) to profile the abnormal behavior in mobile application for user visible failure category which are crash, “stopped unexpectedly” and “not responding”.

The analysis techniques are used in identifying abnormal behavior patterns: 1) To understand the application level activity sequences for abnormal activity via logcat; 2) To identify the objects and classes consuming memory in the java *heap*; 3) To identify system calls or signals made to the OS using Strace. We will discuss in detailed on our framework of analysis techniques in following paper.

By having this framework, it should allow the application developer to conduct investigation and improvement on abnormal behavior application, and hence able to determine the possible caused of “abnormal” application activity in the Android’s application.

References

[1] Sasnauskas, R. and Regehr, J. (2014) Intent Fuzzer : Crafting Intents of Death. 12*th Int. Work. Dyn. Anal.* + *Work. Softw. Syst. Perform. Testing*, *Debugging*, *Anal.*, California, 22 July 2014, 1-5.

[2] Costa-Montenegro, E., Barragáns-Martínez, A.B. and Rey-López, M. (2012) Which App? A Recommender System of Applications in Markets: Implementation of the Service for Monitoring Users’ Interaction. *Expert Systems with Applications*, **39**, 9367-9375. http://dx.doi.org/10.1016/j.eswa.2012.02.131

[3] Chandola, V. (2009) Anomaly Detection : A Survey. *ACM Computing Surveys*, **41**, 1-72. http://dx.doi.org/10.1145/1541880.1541882

[4] Zhao, M., Ge, F., Zhang, T. and Yuan, Z. (2011) AntiMalDroid : An Efficient SVM-Based Malware. *Communications in Computer and Information Science*, **243**, 158-166.

[5] Giuffrida, C., Cavallaro, L. and Tanenbaum, A.S. (2010) We Crashed, Now What ? *Proceedings of HotDep*’10 *Proceedings of the* 6*th International Conference on Hot Topics in System Dependability*, 1-6.

[6] Kim, S., Bettenburg, N. and Zimmermann, T. (2013) Predicting Method Crashes. *Proceedings of* 6*th India Software Engineering Conference*, New Delhi, 21-23 February 2013, 1-5.

[7] Maji, A.K., Arshad, F.A., Bagchi, S. and Rellermeyer, J.S. (2012) An Empirical Study of the Robustness of Inter-Component Communication in Android. 2012 42*nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN*), Boston, 25-28 June 2012, 1-12.

[8] Isohara, T., Takemori, K. and Kubota, A. (2011) Kernel-Based Behavior Analysis for Android Malware Detection. 2011 7*th International Conference on Computational Intelligence and Security* (*CIS*), Hainan, 3-4 December 2011, 1011-1015. http://dx.doi.org/10.1109/CIS.2011.226

[9] Luo, H., He, G., Lin, X. and (Sherman) Shen, X. (2012) Towards Hierarchical Security Framework for Smartphones. 2012 1*st IEEE International Conference on Communications in China* (*ICCC*), Beijing, 15-17 August 2012, 214-219.

[10] Delac, G., Silic, M. and Krolo, J. (2011) Emerging Security Threats for Mobile Platforms. *MIPRO* 2011 *Proceedings of the* 34*th International Convention*, Opatija, 23-27 May 2011, 1468-1473.

[11] Pocatilu, P. (2011) Android Applications Security. *Information Economics*, **15**, 163-172.

[12] Joshi, M. (2012) Analysis and Debugging of OEM's.

[13] Peng, L., Peir, J.K., Prakash, T.K., Staelin, C., Chen, Y.K. and Koppelman, D. (2008) Memory Hierarchy Performance Measurement of Commercial Dual-Core Desktop Processors. *Journal of Systems Architecture*, **54**, 816-828. http://dx.doi.org/10.1016/j.sysarc.2008.02.004

[14] Park, J. and Choi, B. (2012) Automated Memory Leakage Detection in Android Based Systems. *International Journal of Control Automation and Systems*, **5**, 35-42.

[15] Shahriar, H., North, S. and Mawangi, E. (2014) Testing of Memory Leak in Android Applications. *IEEE* 15*th International Symposium on High Assurance Systems Engineering*, Miami Beach, 9-11 January 2014, 176-183.

[16] Wei, X. (2013) ProfileDroid : Multi-Layer Profiling of Android Applications Categories and Subject Descriptors. *Proceedings of the* 18*th Annual International Mobile Computing and Networking*, Istanbul, 22-26 August 2012, 1-12.

[17] Burguera, I. and Zurutuza, U. (2011) Crowdroid : Behavior-Based Malware Detection System for Android. *Proceedings of the* 18*th ACM Computer and Communications Security*, Illnois, 17 October 2011, 15-25.

[18] Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C. and Weiss, Y. (2011) "Andromaly": A Behavioral Malware Detection Framework for Android Devices. *Journal of Intelligent Information Systems*, **38**, 161-190. http://dx.doi.org/10.1007/s10844-010-0148-x

[19] Bl, T., Batyuk, L., Schmidt, A., Camtepe, S.A., Albayrak, S. and Universit, T. (2010) An Android Application Sandbox System for Suspicious Software Detection. 5*th International Conference on Malicious and Unwanted Software*, France, 19-20 October 2010, 55-62.

[20] Karami, M., Elsabagh, M., Najafiborazjani, P. and Stavrou, A. (2013) Behavioral Analysis of Android Applications Using Automated Instrumentation. *IEEE* 7*th International Conference on Software Security and Reliability Companion*, Washington DC, 18-20 June 2013, 182-187.

[21] Kim, H. (2011) MODELZ: Monitoring, Detection, and Analysis of Energy-Greedy Anomalies in Mobile Handsets. *IEEE Transactions on Mobile Computing*, **10**, 968-981. http://dx.doi.org/10.1109/TMC.2010.245

[22] Alazab, M., Monsamy, V., Batten, L., Lantz, P. and Tian, R. (2012) Analysis of Malicious and Benign Android Applications. 32*nd International Conference on Distributed Computing Systems Workshops*, Macau, 18-21 June 2013, 608-616.

[23] Thing, V.L.L., Subramaniam, P.P., Tsai, F.S. and Chua, T. (2011) Mobile Phone Anomalous Behaviour Detection for Real-Time Information Theft Tracking. *CYBERLAWS* 2*nd International Conference on Technical and Legal Aspects of the e-Society*, Guadeloupe, 23-28 February 2011, 7-11.

[24] Guo, C., Zhang, J., Yan, J., Zhang, Z. and Zhang, Y. (2013) Characterizing and Detecting Resource Leaks in Android Applications. 28*th IEEE/ACM International Conference on Automated Software Engineering*, Silicon Valley, 11-15 November 2013, 389-398.

[25] Yan, D. (2013) Systematic Testing for Resource Leaks in Android Applications. *IEEE* 24*th International Symposium on Software Reliability Engineering*, Pasadena, 4-7 November 2013, 411-420.

[26] Franke, D. and Roy, T. (2012) AndroLIFT : A Tool for Android Application Life Cycles. *VALID 2012* 4*th International Conference on Advances in System Testing and Validation Lifecycle*, Lisbon, 8 November 2012, 28-33.

[27] Isohara, T., Takemori, K. and Kubota, A. (2011) Kernel-Based Behavior Analysis for Android Malware Detection. 7*th International Conference on Computational Intelligence Security*, Hainan, 3-4 December 2011, 1011-1015. http://dx.doi.org/10.1109/CIS.2011.226