ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

Study of Various Perspectives of Android Security

[1] Chetan J. Shelke , [2]Pravin Karde , [3]V. M. Thakre

[1]Asst. Professor, Dept.of IT, P.R.Patil College of Engineering, Amravati, India

[2]Asst. Professor, Dept of IT, Govt. Polytechnic, Amravati, India

[3]HOD, Dept.of Computer science SGBAU Amravati University Amravati, India

ABSTRACT: Nowadays, mobile devices are an important part of our everyday lives since they enable us to access a large variety of services. In recent years, the availability of these ubiquitous and mobile services has significantly increased due to the different form of connectivity provided by mobile devices,i.e. GSM, GPRS, Bluetooth and Wi-Fi. Also Smartphone's are steadily gain popularity, creating new application areas as their capabilities increase in terms of computational power, sensors and communication. As various applications are being developed with the added cost of increasing malicious threats. Emerging new features of mobile devices give opportunity to new threats. Android is one of the newest operating systems targeting Smartphone. Several works have recently shown that Android's security architecture cannot prevent many undesired behaviors that compromise the integrity of applications and the privacy of their data.

Currently, there are few secure access control methods of controlling restricted material access on a mobile device but they provides less security. Simple challenge/response protocols do not provide the security required for some restricted information. A lost device or compromised password can easily lead to the compromise of any restricted information to which the device may have access. Due to an inherent portability of a mobile device, the broader, more comprehensive set of access controls is required. Therefore the research work will build a prototype application which will allow an access to restricted information, stored on the devices itself, based on a set of predefined contexts using the device's own hardware.

To overcome the drawback of existing system this proposed work proposes a new approach which is able to perform both static and dynamic analysis on Android programs to automatically detect suspicious applications. Static analysis scans the soft-ware for malicious patterns without installing it. Dynamic analysis executes an application in a fully isolated environment,it intervenes and logs low-level interactions with the system for further analysis. The algorithm can be deployed in the mobile network, providing a fast and distributed detection of suspicious software in a mobile software store akin to Google's Android Market. Additionally, System might be used to improve the efficiency of classical security applications available for the Android operating system.

KEYWORDS : Android Mobile Security, Smartphone, Malwares, Mobile network Services

I. INTRODUCTION

Anti-malware research is recently ongoing process for identifying and analyzing new and unknown malware for extracting possible detection scheme that can be used within some anti-malware software. There exits some malware and malware detector software that can scan and block malwares, Trojans that are infecting Android applications. Most malwares is being detected by scanning in signature database. to generate the reports and special signatures the infected application need to be analyzed and carefully observed so that we can collect some meaningful pattern about the specific malware.

The Android mobile platform is increasing in popularity iOS, Blackberry, Symbian and Windows mobile. It is found that there are currently 675,000 applications in the official Google's Android market, with estimated 25 billion downloads (2012) [1]. At the same time, malware targeting the Android platform has risen sharply over the last two years. According to a report from Fortinet (2011), there exist approximately 2000 Android malware samples belonging to 80 different families [2]. Since discovery of the first Android malware (2010)[3], new families have evolved in sophistication and are becoming increasingly difficult to detect by traditional signature-based anti-malware. More recent families have been observed to exhibit polymorphic behaviors, increased code obfuscation, encryption of malicious payloads, stealthy command and control communication with remote servers. some Android malware like AnserverBot are known to have the capability to fetch and execute payload at run time thus rendering its detection quite challenging. Security experts believe that difficulties in spotting malicious mobile applications results in most Android malware remaining unnoticed for up to 3 months before discovery [2]. Furthermore, Oberheide et al. [4] observed that it took on average 48 days for a signature-based antimalware engine to become capable of detecting new threats there is a need for more effective detection solutions to overcome the aforementioned challenges and mitigate the impact of evolving Android malwares.

Both the iOS and Android mobile OS implement some form of application privilege separation model, which means the underlying kernel isolates each application into an execution sandbox that shields the application from unauthorized access to its data [5]. An application makes access requests to gain the necessary permissions to protected resources— these permissions are granted either by the kernel or explicitly by the user. One central problem, is the inability of users to make good security choices. As studies show, typical users have neither the necessary understanding of the available security mechanisms nor the ability to properly utilize those protection mechanisms to their full benefit [6]. For example, users have little trouble granting dangerous permissions when installing free Android applications. The burden placed on users for managing arcane security options provides an opportunity for attackers to take advantage of the gap between the perception and reality of risk. To compound the problem, mobile applications originate from nearly as many developers as the number of mobile applications onto the market. Unlike traditional commercial desktop software, in which most of the publishers are known and reputable companies, every mobile app developer is a publisher. Mobile app stores are distribution channels that can make an application available to the entire community seemingly instantaneous, with less or no security review.

II. OVERVIEW ON ANDROID

Android implements a complete software stack for running mobile device applications. At the lowest layer is a Linux kernel that provides device drivers, memory management, power management, and networking. Next, Android provides some native libraries for graphics, database management and web browser functions, which can be called through Java interfaces. Furthermore, Android includes core Java libraries, and a virtual machine for running customized byte code (called dex) derived from JVML byte code. Above this layer is the application framework, which serves as the abstract machine for applications. Finally, the top layer contains application code, developed in Java with the APIs provided by an SDK.

An Android application is a package of different components, where each of which can be instantiated and run as necessary (possibly even by other applications) [7]. Components are of the following types:

- Activity components form the basis of the GUI. Basically, each window of the application is controlled by some activity.
- Service components run in the background and remain active even if windows are switched off. Services can expose interfaces for communication with other applications.
- BroadcastReceiver components react asynchronously to messages from other applications.
- ContentProvider components store data relevant to the particular application, usually in a sqlite database. Such data can be shared across applications.

Android is an OS designed for smartphones. Depicted in Figure 1, Android provides a sandboxed application environment for the execution applications. A customized embedded Linux system interacts with the phone hardware and an off-processor cellular radio. The Binder middleware and application API runs on top of Linux. To simplify, an application's only interface to the phone is through these APIs. Each app is executed within a Dalvik Virtual Machine (DVM) running under a unique UNIX UID. The phone comes pre-installed with a selection of system applications, such as phone dialer, address book.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

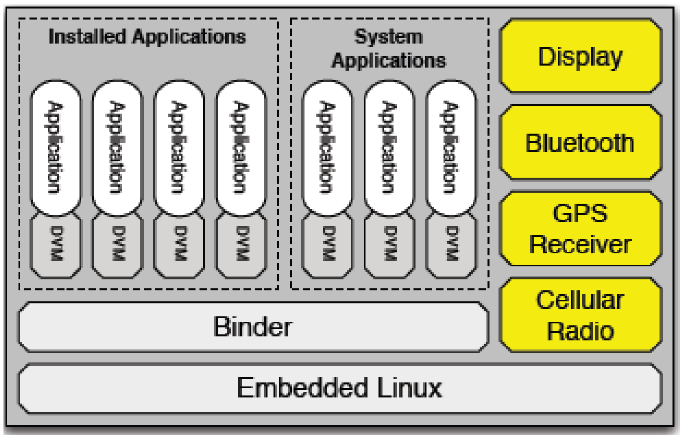

Figure 1: The Android system architecture

Applications interact with each other and the phone through different forms of IPC. Intents are typed inter-process messages that are directed to particular systems services or applications or broadcast to applications subscribing to a particular intent activity type. Persistent allow provider data to stores queried through SQL-like interfaces. Background services provide callback and RPC interfaces that applications use to access data or trigger actions. In last user interface activities receive named action signals from the system and other applications.

Binder acts as a mediation point for all IPC. Access to system resources (e.g., GPS receivers, text messaging, phone services, and the Internet), data (e.g., address books, email) and IPC is governed by permissions assigned at install time. The permissions requested by the application and the permissions required to access the application's interfaces/data are defined in its manifest file. To simplify, an application is allowed to access a resource or interface if the required permission allows it. Permission assignment and indirectly the security policy for the phone is largely delegated to the phone's owner: the user is presented a screen listing the permissions an application requests at install time, which they can accept or reject.

The Android Security Policy is divided into groups; "Interaction Policy" and "Permission Granting Policy". Signaturebased Policy, Protection Level-based Policy and Application Configuration-based Policy are found during installation in the "Permission Granting Policy"and Interaction Policy covers four different types of policies,

1) Permission based Access Control Policy,
2) Signature based Policy (SBP),
3) Application Configuration based Policy (ACBP),
4) Context-based Policy (CBP).

Figure 2 shows the different security policies of the android OS. Interaction Policies are defined at runtime, for example Signature-based Policy can be used to restrict the component applications. The implementation is based on the applications' signatures, which includes default-allow and default-deny modes [8].

Android Security Architecture

Android is build on Linux kernel 2.6. The security architecture and permission model are therefore same on Android as that on linux OS. This makes Android to be a multi processing system, and running each application in its own separate process. Security is implemented at process level in Android. It implements security procedures through different mechanisms at different levels. This includes implementing security at application level through user and group ID's. At component level it makes use of permission mechanism to restrict access to specific component while at data level it implements security through per URI basis permissions. Android architecture is defined such that no application can perform an operation on any other application, its components or its data, such as, Reading and/or writing. This restricts access to Private data (such as contacts or e-mails), another application's _les, performing network access, keeping the device awake etc [11]. The only way to get access to any component or data is to explicitly declare the permissions it needs for that specific additional capabilities. Application level security is discussed in this chapter, while components and data level security is discussed in followed chapter.

Application Level Security

- Security implemented by Application Signing using Certificates mechanisms

The author of application is identified by a certificate. This certificate is signed by the developer of that particular application, whose private key is held by the same person. The certificate impacts the security in the following way: To establish a trust full relation between different applications. Who can access the signature based permissions. Who can share the user id.

- Security enforcement using USER ID

As Android is based on Linux kernel, so the security policy enforced is same as that implemented for Linux based system. At the time of installation of an application on Android, its assigned with a unique user ID. This user ID will be remain same throughout the application life time. Assigning User ID prevents applications to access other applications or their data, as the data associated with a particular application is assigned the ID of that application.

III.PROBLEM DEFINATION

One big benefit of shifting the security functionality into the mobile network is the almost indefinite processing power and “battery” capacity. This makes it possible to run very resource intense security services that would not be feasible on the phone. If the phone is replicated in the mobile network, this also allows the developer of a security service to extend this service without changes on the phone. The security service can examine the phone not only from inside its system (similar to an application on the phone), but it can also monitor the replica itself which runs the mobile network (e.g. look at the connections the replicated phone attempts to make). This can further improve the chances of finding malicious software and open up possibilities that would not be feasible on the device itself. For example, detecting a root kit could be impossible on the phone itself, but a security service which only scans the replica’s files without executing the replica, might be able to detect the root kit. But the shifting of the security functionality into the mobile network could also be problematic, if not all parts of the phone can be replicated into the mobile network. Previous work has shown that it is possible to run smartphone applications in the mobile network (e.g. [21]).

Smartphones typically possess only a limited amount of battery capacity and processing power. Once the security functionality is applied outside the physical device, these limitations can be circumvented.

Emulating the smartphone in the mobile network comes with one problem: every different smartphone needs its own emulator to be perfectly emulated. Since the phones typically differin certain aspects (e.g., operating system, available sensors, etc.), no emulator can simulate more than one smartphone perfectly. To support multiple devices, many different emulators need to be implemented. Another approach would be to implement a more generic emulator which can support more than one device. This is the case for Android devices where one emulator can represent many different devices, with the downside that these devices are not emulated accurately.

Since the purpose of the security system is the detection of malicious applications, such applications do not pose a security threat per se. A malicious application can only influence the security system if it tries to manipulate the system itself, for instance, by modifying the generated images or user interactions. To prevent this, these files must be stored in a secure way, and the integrity of the files and the security system’s components on the smartphone and inside the mobile network have to be guaranteed. The integrity can be guaranteed by implementing cryptographic principles (e.g., by using file checksums, or by signing the components using public key cryptography [36,46]). If the integrity is not guaranteed, the security service’s announcements become useless, since a malicious application can hide itself from detection (e.g., by not copying itself into the images), or trick the security service into reporting other applications as malicious, even if they are not. Since the components of the security service for different smartphone users do not influence each other, a compromised smartphone can not disrupt the whole security system

At present there exists several systems for detecting and avoiding malwares and malwares available in android applications but apart from this there are some shortcomings presents in these system which are

1. Existing malware detection mechanism are very time consuming process as each time consuming process as each time a new application downloaded have to be checked and analyzed for behavior patterns and datasets are recorded and stored.
2. Extremely worst battery performance as Antimalware application and other processes has to be run in background.
3. There are several approaches of detecting malwares. Anomaly detection is one of them in which various behavior patterns are observed and stored in various datasets. The main drawback of this is that, one have to maintain large datasets and as no. of datasets increases then there also exists problem of inconsistency and redundancy.
4. Existing system Crowroid which detects Trojans like malwares on Android using analyzing no. of system calls each time is executed.

IV.PROPOSED MODEL

This proposed work will introduces an architecture for a security system, which will detects malicious applications for smartphones. The detection of these malicious applications is not done on the smartphone itself, but in a different environment, for instance, a mobile network, where the detection algorithm is applied to a virtual smartphone. The whole system containing all parts is called the Security System, the part in the mobile network is called Security Service. There are various reasons for the domain change from the smartphone to the mobile network: Applying the security functionality on a virtual device and not on the real smartphone offers the possibility to run analysis that would be problematic otherwise. For a normal smartphone, which is managed by an customer, it might be infeasible to install a custom firmware, or modify parts of the operating system for certain checks, which has to be done e.g., for taint checking . The virtual device does not have these barriers, it can be changed arbitrarily for every malware detection concept as needed. The virtualisation of the smartphone also has the benefit of posing no risk to real hardware. If the virtual device gets damaged, e.g. by a malicious application, it can just be recreated without any cost. Another benefit of such a security system is the centralization of the malware detection. In addition, they are able to accurately decide what to check and monitor the results. This process can even be done on a daily basis.
Android has more than 100 permissions, and broadcasts, thus it gives us a probable number of combinations of permissions and broadcasts which can be saved as malware definitions and later used for comparison with application signatures, which would help us to separate the malicious applications from the good ones.

Once the applications are filtered then malicious applications will be reported on to the mobile network so that everyone using these secure application can be informed about the malware applications which are installing or already installed on their devices. This technique would work for applications which are being installed on our devices or are being installed on our devices. Once detected the filtered applications are removed from our device and our phone is free from malwares. In this approach we can also use a Smart Agent, which would be a dummy malware, and try to access sensitive data on our phone, then trace its own behavior while accessing the data. Once the behavior is traced, then this behavior can be compared with other programs to track if they are also accessing data in a similar manner, thereby marking them as malwarees and reporting them over the mobile network. Thus making our system more robust and prone to malwarees and malwares.

V. CONCLUSION

This paper has attempted to review a significant number of papers to cover the recent development in the field of Android security. In the review paper first we try to give the all information about Android mobile, its structure and Operating System. We also give brief overview about the Android application and its functionality. The security policies of android are discussed in the paper with its structure. We also take a overview on the different security mechanism currently used on android mobile for the detection of malwares application from the mobile. The list of references to provide more detailed understanding of the approaches described is enlisted. We apologize to researchers whose important contributions may have been overlooked.

REFERENCES

[1] M. Becher, F. Freiling, and B. Leider. “On the effort to create smartphone worms in windows mobile”, Information Assurance and Security Workshop, 2007. IAW ’07. IEEESMC, pages 199–206, 20-22 June 2007.

[2] J. Burns. “Developing secure mobile applications for android - an introduction to making secure android applications”,https://www.isecpartners.com/files/iSEC_Securing_ Android_Apps.pdf, 2008. [Online; accessed 05-Sep-2013].

[3] W. Enck, M. Ongtang, and P. McDaniel. “Understanding android security”, IEEE Security and Privacy, 7(1):50–57, 2009. [Online; accessed 02-Sep-2013].

[4] C. Mulliner. Advanced attacks against pocketpc phones. 2006. [Online; accessed 04-Sep-2013].

[5] C. Mulliner. Exploiting symbian: “Symbian exploitation and shellcode development”, http://mulliner.org/ symbian/feed/CollinMulliner_Exploiting_ Symbian_BlackHat_ Japan_2008.pdf, 2008. Talk on BlackHat Japan 2008, visited 15.6.2009.

[6] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C.Wolf, “Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices” ,in Proceedings IEEE Security and Privacy, May 2011. [Online; accessed 04-Sep-2013].

[7] “R. G. Chicone, An Exploration of Security Implementations for Mobile Wireless Software Applications within Organizations. Minneapolis: Graduate Faculty of the School oBusiness and Technology Management, Northcentral University, 2010.”

[8] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, “Heterogeneity in mobile cloud computing: Taxonomy and open challenges,” IEEE Communications Surveys Tutorials, vol. Early Access Online, 2013.

[9] Guo, M.; Liaw, H.; Hsiao, L.; Huang, C.; and Yen, C., "Authentication using graphical password in cloud", Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on , pp.177-181, 24-27 Sept. 2012.

[10] Khitrov M., "Talking passwords: voice biometrics for data access and security", Biometric TechnologyToday, Volume2013, Issue2, February2013, Pages9-11, ISSN 0969- 4765, http: //dx .doi .org /10. 1016/S0969-4765(13)70036-5.

[11] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, December 2005.

[12] D., M'Raihi, S., Machani, M., Pei and J., Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, May 2011.

[13] Chen, T.; Yeh, H. and Shih, W., "An Advanced ECC Dynamic ID-Based Remote Mutual Authentication Scheme for Cloud Computing", Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on, pp.155-159, 28-30 June 2011.

[14] Yassin, A.A.; Hai J.; Ibrahim, A.; Weizhong Q. and Deqing Z., "A Practical Privacy- preserving Password Authentication Scheme for Cloud Computing", Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, pp.1210-1217, 21-25 May 2012.

[15] Jaidhar, C.D., "Enhanced Mutual Authentication Scheme for Cloud Architecture", Advance Computing Conference(IACC), 2013 IEEE 3rd International, pp.70-75, 22-23 Feb. 2013 doi: 10.1109/IAdCC.2013.

[29] D. Mohammed, “Security and Cloud Computing: An Analysis of Key Drivers and constraints”, Information Security Journal: A Global Perspective, vol. 20, (2011), pp. 123-127.