ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (706-713), Month: April - June 2015, Available at: www.researchpublish.com

MOSES: Supporting and Enforcing Security Profiles on Smartphones

[1]Ramkrishna. G. S, [2]Dr. B R Prasad Babu, [3]Shaik Mohammed Ghouse

Abstract: **To increase the productivity of business users Smartphones are very effective tools. In smartphones the end users can perform several tasks and they can always be updated because of their increasing computational power and storage capacity. As there is the increase in productivity of employees the companies are willing to support the employee-owned smartphones. The issues about security concerns such as Data loss, Data Sharing and Data leakage have reduced the usage of smartphones for companies. In this paper, we introduced Moses, which is a policy based framework to enforce the isolation of application by using the software and the data in the android platform. Through MOSES it is possible to show different Security Profiles in a individual smartphone. Every security profiles consists of an policies which controls the accessing of data and applications. Where these profiles are not hardcoded or predefined because of which it can be used at any time. One important role of the MOSES is to dynamically switch between one security profile to another security profile.**

Keywords: **BYOD, Security profiles, Android, Para-virtualization, Context.**

1. INTRODUCTION

Throughout the worldwide the sales of smartphones is up to 455 millions and there is a 46 percent of increase from the year of 2012 to 2013. There is 77 percent increase in smartphones and 11 percent increase in mobile phones. As the number of users increased to use the smartphones and as the average selling price of the smartphones is almost relatively similar to the old featured phones so there was gradual decrease in the use of old featured phones. In the smartphones world the most used OS is an Android OS where as the Android OS has an 82 percent share in market [1]. The above figures shows the easily acceptance of Android because of its easy to used by third party developers.

Many tasks can be performed by the end users at a time. So the end users need their own smartphones to work with their IT company. Now-a-days in many mobiles the desktop version is available. According to the detailed study the employee productivity can be increased by allowing the access to the enterprise services by using the smartphones. So the employees of an company using their own devices so the company has brought a BYOD Bring Your Own Device policy [2], allowing the employees to mobile access the company's application by using their smartphones. Many mobile device manufacturers have started to produce the device to handle dual subscriber identification modules (SIM's) simultaneously.

Because of this advantage the end users can use third party applications, but there arises a security threat such as malicious applications may gain the access to emails, MMS and SMS which is stored in the smartphones which contains the secured data of the company. And there are some more threats like leakage of data which are not so necessary for the functions of applications to users. So these are the issues for the company to protect the data that are used by the users from such attacks.

To overcome this problem one of the possible solution is Isolation, that is by arranging the applications and data according to the private or personal data or the recreational applications are separated for work. To provide the security on the same device we might separate the security environments. One security environment can be used for corporate data and trusted applications and another security environment can be used for third party purpose such as any popular applications like games, social network sites etc. So as long as it not possible for second environment to access the applications/data from the first environment there will be low risk of leakage of confidential information.

This solution can be implemented by the technology known as **Virtualization**, where in this the different OS can be installed and used on same device for the work. Virtualization is good to use in the PCs and servers where as for an embedded systems such as smartphones it is more resource demanding. So the another approach is the paravirtualization which is the less resource demanding which is suitable for full fledged devices such as PCs and also embedded systems like smartphones.

Like in the virtualization where as the guest OS is not knowing of running in virtualised environment, where in paravirtualization [4] the guest OS is modified to increase the performance. In smartphones the paravirtualization is still under development and there still need to be exist many solutions. There are the limitations for the solutions of virtualization that is the coarse grained approach (such as this environments are used separately when there might be a interaction). And another limitation is hardcoding of environment specification. Because the problem is, the user or the company cannot define the environments because this environments are predefined or hardcoded already in the virtual machine. So the switching among these environments requires the user interactions where it can take more amount of power and time. Whereas the researchers are trying to improve some of these aspects but this environment separation of virtual machine is impossible to adapt or change of their specifications where it will remain as an open issue.

1.1 Contribution:

In this paper we introduce the MOSES, whereas the solution of separating the mode of use in smartphones. Where as in MOSES it implements the soft virtualization through the isolation.

- The one or more contexts are associated with the security profile by which it can be determined when this profile become active. Where these contexts are differentiated as high level features (trust level, reputation, etc) and low level features (time and location).
- End users can easily and dynamically specify the context and profiles. The GUI is used for this purpose which is provided by the MOSES, where these profiles can be differentiated to the level of single application and single objects.
- To switch between the security profiles where it requires the user interaction or it can be automatic, transparent etc.

By using the MOSES we ran a thorough set of experiments to check its effectiveness and efficiency. By these experiments the solution which we provided can be accepted for energy and storage consumptions where it is feasible.

The smartphones of today's fail to provide the information of how to gain control over and visibility where this third party users uses the applications for their private data. So to overcome we came up with an traintdroid which is an system wide dynamic tracking capability where it can track multiple sources of data. By defining Android's virtualised execution environment the traintdroid also defines the real time analysis.

By using the traintdroid to check the behavior of the third party android applications we chosen 30 third-party android applications, we found 78 instances of misuse of users private information over the 18 applications. Using the TraintDroid to monitor the sensitive data which is used by the third-party applications to identify the misbehaving of applications.

2. ANDROID SECURITY

Open Handset Alliance (OHA) developed the Google Android [5] which is a Linux based platform [6]. The Android applications can also be programmed in Java and where it is compiled into a bytecodes which is run by Dalvik Virtual Machine (DVM). Whereas Android package has its own address space and it is executed in its address space and in a separate DVM. By using the any one of the following basic components Android applications can be built. *Activities* defines the user interface, *Services* in which background processes are executed, *Broadcast Recievers* these are the mail boxes to provide the communication between the components of the same application or to the different application, *Content Providers* it is used to share and store the applications data. And these applications components communicates through messages which is called as intents.

Android [7] Keeping developers in mind it is developed. To reduce the burden on developers security controls were designed. Flexible security controls provides to security-savvy developers to easily work and rely on them. The attackers attempts to attack on users applications by such as social engineering attack where in this they try to convince users to install malware and they can also attack to third-party applications. In this paper we discussed the security features of Android platform of specific applications, which are those related to the SMS or browser application.

Following are the Android platform building blocks [8]:

- **Hardware Device**: To run Android we need a wide range of hardware configurations such as tablets, smartphones, set-top-boxes etc. Android which is a processor-agnostic and it also takes advantage of hardware-specific security capabilities which includes ARM v6 execute-Never.
- **Android Operating System**: On top of Linux Kernel the core operating system of Android is built. We can access all device features which includes GPS data, camera functions, Telephony functions, Bluetooth functions, network connections, etc through the operating system.
- **Android Application Runtime**: Usually Android applications will be written in the Java programming language and executed in virtual machine called Dalvik. Android applications are native applications(or may include native libraries). One example of such applications is Android services. Dalvik and native applications run in the same security environment, which is contained in the Application Sandbox. For Android applications there is a fixed part in the file system. They can write private data such as including databases and raw files.

Android applications can also extend the core Android operating system. Hence we are having to discuss about primary sources of applications:

- **Pre Installed Applications**: There are a set of pre-installed applications which Android includes such as contacts, calendar, web browser, phone and email. The pre-installed applications functions as user applications and as well as it provides key device capabilities which can be accessed by other applications. These applications can be developed with OEM for some specific device or it may also be a part of the open source platform of Android.
- **User Installed Applications**: An open development environment is provided by Android which supports all or many of the third-party applications. Google Play offers to their users hundreds of thousands of applications.

A set of cloud-based services are provided by Google which are available to work on any compatible Android device. Some of the main services are:

- **Google Play**: Google Play is basically a collection of services which allow users to download, install, and purchase applications from the internet on Android device or the web. Through Google play it is easy to developers to reach Android users and customers. Many of the Google Play asks for application license verification, community review, application security scanning, and also other security services.
- **Application Services**: Use cloud capabilities such as data back up and settings for push messages are allowed by Android applications in some Frameworks.
- **Android Updates**: The Android update services is having capabilities of security updates to Android devices, which includes the updates through internet or over the air (OTA).

Considering on security, Android consists two levels of enforcements, the application framework level and at the Linux Kernel level: At the application framework level the Androids provides the success control through the Inter Component Communication reference monitor. Whereas this reference monitor provides Mandatory Access Control (MAC). The Android is a multi process system to the Linux kernel level. The unique Linux user identifier (UID) and Group Identifier (GID) are used to an application during installation. In Linux for users there are three types of file access permissions: the owner of the file, the users of same group and the other users.

Android is an open source operating system for mobile devices which provides a base operating system along with an application such as Java software development kit (SDK) and a middleware layer, and a collection of system applications. Even though Android SDK is available since 2007, which is the first Android that was available, many new products and applications are also now available for it. Android lets developers limitlessly extend online services to phones. It is Androids chief selling point. All Android users simply provides a username and password, and their phones will automatically get synchronized with Google services [9].

3. RELATED WORK

In this section we discuss about the overview of the related work. In section 3.1 we will discuss about extending the security of the Android platform. In section 3.2 we discuss about the BYOD for the mobile systems. Section 3.2.1 consists of Solutions based on secure container (SC). Section 3.2.2 consists of the virtualization solutions concept. In section 3.2.3 consists of the solutions which adopted other approaches.

3.1 Android Security Extensions:

In order to improve the security of Android, lots of solutions have been proposed. Here we consider the solution which is related to our system.

In Android, during the installation time, the permissions requested by the users are granted in manifest file. In Android, all-or-nothing approach is supported, which means the user should provide grant for all the permissions requested or should abort the installation of the application. Revoking the permission during the runtime is not possible. To overcome this coarse-grained approach many solutions have been proposed. Sometimes the users need to grant the permission to the application during the installation time, for that we use Apex [10]. Saint [11] which is a policy based application management system which aims at controlling the way in which the applications interact with each other. To automatically control granting the permission during run time CREPE [12] is used which allows the user to create the policies.

MockDroid is a system used to protect the users private data by limiting the access of the installed applications by filtering out the information on phone data. TaintDroid is used to dynamically analysis to control the how data flow between application happens. TaintDroid are associated with the predefined data source, such as contact book, SMS, device identifier (IMEI), etc. It also does tracking of the flow of Tainted data and it also notifies for users whether the Tainted data leave the device from the outbound network connections.

3.2 Bring Your Own Device Approaches:

To provide the security for the Android some solutions are aimed to support the BYOD. Some of the important solutions are as follows:

***3.2.1 Secure Container*:**

In the phone at the application layer the secure container creates an isolated environment where secure container is a special mobile client application, where the enterprise administrator creates the policies which control this isolated environment but it cannot control the behavior of a user outside the container by using the security container.

Security container can be implemented as a user applications for many commercial solutions where Nitro Desk Touch Down [13] and Good [14] offer solutions in the container.Whereas to develop new applications the other solutions such as Fixmo [15] offers a set of basic applications.

3.2.2 Mobile Virtualizing:

To distinguish the hardware from the OS point of view are isolated by using virtualization. The virtual machines activities and for guaranting such isolation is done by hypervisor. Whereas in traditional computers virtualization is widely used for: (i) reduce the cost of deployment of applications; (ii) increase security.

Using of mobile device widely increased so to increase the performance of the device porting the virtualization to the mobile platform is became actual. Advantages of virtualization in mobile system are like: (i) It can be used to separate the communication subsystem from high level application core; (ii) It can also be used for license separation; (iii) The communication stack security can be increased. There are several barriers to adopt the virtualization in mobile devices.

3.2.3 Other approaches

As we discussed the above approaches there are some more solutions. Gupta et al [16] to support the dual mode of operation, enterprise and private he modified Android Framework. Where this modifying restrict the use of phones communication capabilities and it force the communication through enterprise VPN. And it also consists the encrypted external storage in enterprise mode. To separate the data exchange between different domains the authors of TrustDroid [17] proposed to monitor the IPC communications, network traffic and file system access.

Where the other solutions to track the sensitive information they use the capabilities of TaintDroid. One of the main use of this approach is how to discover the sensitive information. Where Feth and Jung [18] proposed to depend on external authorities which supplies data usage policies with data.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (706-713), Month: April - June 2015, Available at: www.researchpublish.com

4 MOSES OVERVIEW

MOSES Modes Of Uses Separation In Smartphones is the approach used in this section which describes briefly about MOSES:

The application and data installed in a single device are separated by using abstraction which is provided by the MOSES, such as the personal data and application is separated by corporate data and application. The data and applications are stored in compartments which are provided by our approach. The data and apps stored in one compartments are isolated by other compartments where this is guaranteed by the MOSES enforcement mechanism. Where in MOSES this compartments are called as security profiles where data and apps are stored. Generally the security profiles is a set of policies which defines what application can be executed and what data to be accessed.

The automatic activation of security profiles are done based on context where this feature is introduced in MOSES. One context may be associated with one or more security profiles, the context are the Boolean expression where the information are gathered from the Smartphone Raw sensors (eg., GPS sensors) and Logical sensors. Logical sensors are used to capture the user behavior by combining the raw data from physical sensors. The SP is activated when the context definition is evaluated to true for which the SP is associated to that context definition. When the several context are associated to the different SP's it is possible that it might all active at the same time. So to avoid such conflicts in MOSES the SP's are associated with priority to the MOSES, the highest priority assigned to the SP is activated first, if SP's have the same priority then the SP which is activated first will remain active.

In MOSES the user can switch to specific SP and where as the user can enforce the MOSES to activate the specific SP through the system apps provided by the MOSES. Each SP is protected by using the password where they are associated with an owner of the profile. In an device the SP can be created /edited by using the app installed in the device. Additionally the remote SP management is supported by the MOSES, formally the users managed their SP by phones, where now a days the enterprise administrator are there to control work SP. Where the user might tamper the work SP so to avoid that the security administrator protects it by password.

The idea of lightweight separation of SP is implemented in the current version of the MOSES although the same idea is exploited but the approach used by the MOSES is new. To split the data between the different profiles the previous version of MOSES is depended on the TaintDroid. Where as in the current version of the MOSES the data separation takes place by Linux kernel level through the filesystem approach. Whereas this provides our app data segregation out of the box. If user want to apply fine grained constraints to data then only user can define the security policies in the new version of the MOSES.

5. ARCHITECTURE

MOSES consists of the components as shown in the figure.1. The notion of context is presented in center of the MOSES. Detecting the context activation/deactivation is the responsibility of the component i, the role of the *ContextDectectorSystem* is notify to the *SecurityProfilemanager* if any such event occurs.

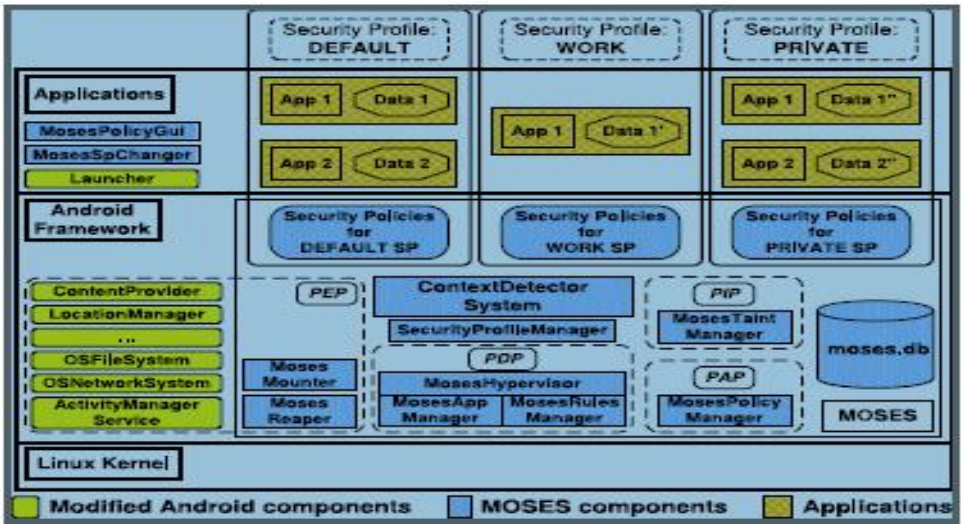

FIG 1: MOSES Architecture

The information of the SP whether it is linking to one or more context is handled by the *SecurityProfileManager*. Whereas the activation and deactivation of an SP's is the responsible of i. The *SecurityProfileManage*r implements the following logic:

- The notifications is ignored when the new activated context is corresponded to active SP.
- A newly active context if it has lower or equal priority to the currently running SP then notification is ignored.
- Whenever the SP switch happens the old SP is deactivated and new SP gets activated which can be done in all cases.

The *SecurityProfileManager* informs to the *MosesHypervisor* which is the new SP's that need to be activated by sending the command.

In MOSES, the *MosesHypervisor* is a component which acts as a policy decision point (PDP). The *MosesHypervisor* is a central point for MOSES security cross checks the policies for the active SP access to resources. The *MosesHypervisor*'s policy checks for its two managers: (i) The *mosesappmanage*r; (ii) The *mosesrulesmanager*, and then they decide which applications are allowed to be executed within a SP and later they take care of managing special rules.

In *TaintDroid* the taint values are stored in "shadow database", it is managed by the component called as *MosesTaintManager*. The enforcement of the policies are dependent on the information provided by the *MosesTaintManager*, and this component acts as policy information point (PIP).

Policy enforcement point (PEP) need to be enforced by *MosesHypervisor* decision. The Several components within android middleware are affected by Moses because of decisions are pending for enforcing. So only the, PEP has several components which offering the system services such as *LocationManagerService* and *ActivityManagerService*. In the new SP after the switch the process of applications need to be shut down. It is the responsibility of the *MosesReaper*. *MosesMounter* provides the functionality to separate the data between profiles and different file system.

The two MOSES applications such as *MosesSpChanger* and the *MosesPolicyGui* allows the user of the device to interact with MOSES. User can manually activate the SP by using the *MosesSpChanger*. User can manage the SP's by using *MosesPolicyGui*.

6. IMPLEMENTATION

In this section implementation of some key aspects of MOSES are described. Android Open Source Project (AOSP) is used to define the versions in MOSES version 2.3.4_r1.

6.1 Context Detection:

MOSES can automatically switch the SP's based on their current context, this is one of the major contribution of the MOSES. The activation and deactivation of the context is notified to the listeners by the *ContextDetectorSystem*. One of this listeners is a *SecurityProfileManager* notifies about the change through the callback functions onTrue and onFalse, which can be activated and deactivated of a security context respectively.

6.2 Filesystem Virtualization:

Directory polyinstantiation is a technique used to separate data between different SP's. According to some system parameters a Polyinstantiated directory provides the different instances of itself. Android creates its home folders and assigns it to Linux file permission to allow to the owner of the directory to access the data stored in it during the installation of an application. Because of the use of some applications is prohibited in some SP's soothe formal application allows MOSES to control direct access to the physical directories while to decrease storage over head.

6.3 Dynamic Application Activation:

As soon as this profile is activated a list of application UID's which are allowed to be run, are provided to each SP. The application receives its own UID during its installation and these identifier are used by MOSES to control which application can be activated for each SP. The same UID can be shared by some packages. The Android system assigns the same UID to these applications during the installation time. The *MosesAppManager* list the UID's and selects from the MOSES database, which are allowed in the activated profile and stores it into the allowed UID's.

6.4 Attribute Based Policies:

An attribute based access controls (ABAC) model [19] is enforced by MOSES within each SP. To constraint application behavior users defined the fine-grained access control policies within each SP. To define and edit policies MOSES provides an activity as shown in figure.2.

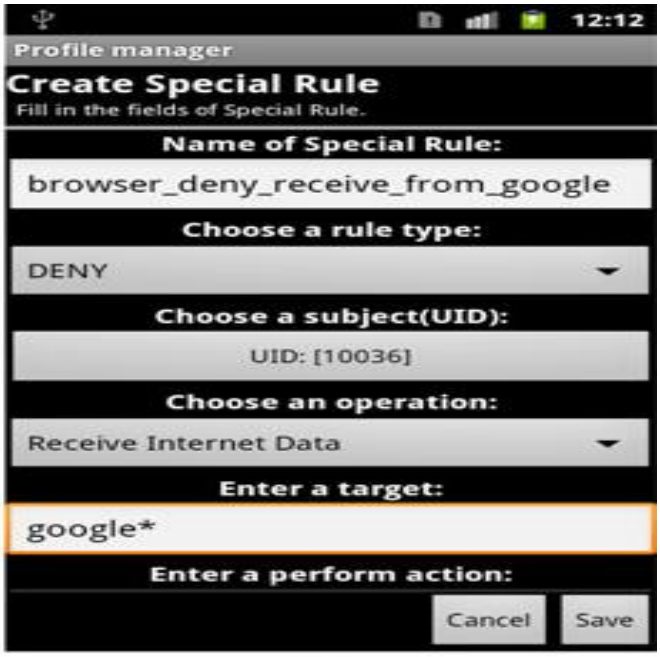

FIG 2: ABAC rule creation

7. CONCLUSION AND FUTURE WORK

MOSES provides the policy-based security containers implemented via software. To prevent the application to be able to bypass our isolation at the system level. However, MOSES has some limitations: First, using the UID we specified the fine-grained policies and allowed applications. But it is possible that some applications share the same UID in Android. Thus, by applying the MOSES rules and restrictions to one applications that automatically will be extended to the other applications with same UID. We can bypass MOSES protection by using the root access of the application. Because of which MOSES is ineffective in defending the malwares that obtains to root access, eg., rootkits.

MOSES can be improved in several aspects, such as to make easier policy-specification process, i.e the solution is to embed into the system policy templates where we can simply select and associate to an application. And also that currently MOSES does not separate the system data and information's on SD cards. So in the future we are planning to add this functionality in the system, and to reduce the performance overhead in the future.

REFERENCES

[1] According to Gartner the survey of Smartphone use increased in 2013, http://www.gartner.com/newsroom/ id/2623415, 2014.

[2] Bring Your Own Device (BYOD) Policy is established by the Unisys, http://www.webopedia.com/TERM/B/BYOD .html/unisys_establishes_a_bring_your_own_device_byod_policy2014.

[3] TaintDroid is an information flow tracking system for monitoring on smartphones http://appanalysis.org/.

[4] Para Virtualization performance is evaluated by modern phone platform http://www.vmware.com/ files/pdf/ VMware_paravirtualization.pdf.

[5] GOOGLE ANDROID it is a comprehensive assessment published in https://www.android.com/

[6] Android, http://www.android.com/ 2014.

[7] Google Android Security: https://static.googleusercontent.com/media/source.android.com/ en//devices/tech/ security /reports/Google_Android_Security_2014_Report_Final.pdf

[8] Android platform building blocks.

[9] css.csail.mit.edu/6.858/2014/readings/android.pdf.

[10] Extending Android Permission Model the User Defined Runtime Constraint are enforced,": Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS '10), pp. 328-332, 2010.

[11] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android," Proc. Ann. Computer Security Applications Conf. (ACSAC '09), pp. 73-82, 2009.

[12] A System is Enforced Fine-Grained Context Related Policies on Android," .M. Conti, B. Crispo, E. Fernandes, and Y. Zhauniarovich, "CRePE: IEEE Trans. Information Forensics and Security, vol. 7, no. 5, pp. 1426-1438, Oct. 2012.

[13] The Nitro Desk Touch Down is related paper, http://www.nitrodesk.com/TouchDown.aspx, 2014.

[14] The Good BYOD Solutions in this paper, http://www.good.com/mobility-management- solutions/bring-your-own-device, 2014.

[15] Fixmo of the SafeZone: Corporate Data Protection, http://fixmo.com/products/safezone, 2014.

[16] A. Gupta et al., A. Joshi, "Enforced Security Policies in Mobile Devices Using Multiple Personal," Proc. Seventh Int'l ICST Conf. Mobile and Ubiquitous Systems: Computing, Networking, and Services (MobiQuitous), pp. 297-302, 2010.

[17] The ACM workshop security and privacy in smartphones and mobile devices, PP.51-62,2011.

[18] D. Feth and C. Jung, "Context Aware, Data Driven Policy Enforced for the Smartphone Mobile Devices in Business Environment," Proc. Int'l Conf. Security and the Privacy in the Mobile Information and Communication Systems (Mobi Sec '12), pp. 69-80, 2012.

[19] The attribute based access control (ABAC); http://www.axiomatics.com/attribute-based-access-control.html