MOSES: SUPPORTING AND ENFORCING SECURITY PROFILES ON SMART PHONES

Vemula Divyasri
UG Student, Bojireddy College of Engineering for women, Saidabad ,Hyderabad, Telangana India

Prof.V.V.R.L.S.Gangadhar
Professor, Princeton College of Engineering and Technology, Hyderabad, Telangana,India

Abstract:

Smartphones are very realgears for cumulative the productivity of business users. With their increasing computational power and storage capacity, smartphones allow end users to perform several tasks and be always updated while on the move. Companies are willing to support employee-owned smartphones because of the increase in productivity of their employees. However, security concerns about data sharing, leakage and loss have hindered the adoption of smartphones for corporate use. In this paper we present MOSES, a policy-based framework for enforcing software isolation of applications and data on the Android platform

Introduction

Smartphones are very effective tools for increasing the productivity of business users. With their increasing computational power and storage capacity, smartphones allow end users to perform several tasks and be always updated while on the move. Companies are willing to support employee-owned smartphones because of the increase in productivity of their employees. However, security concerns about data sharing, leakage and loss have hindered the adoption of smartphones for corporate use. In this paper we present MOSES, a policy-based framework for enforcing software isolation of applications and data on the Android platform. In MOSES, it is possible to define distinct Security Profiles within a single smartphone. Each security profile is associated with a set of policies that control the access to applications and data. Profiles are not predefined or hardcoded, they can be specified and applied at any time. One of the main characteristics of MOSES is the dynamic switching from one security profile to another. We run a thorough set of experiments using our full implementation of MOSES. The results of the experiments confirm the feasibility of our proposal.

EXISTING SYSTEM:

A solution could be implemented by means of virtualization technologies where different instances of an OS can run separately on the same device. Although virtualization is quite effective when deployed in full-fledged devices (PC and servers), it is still too resource demanding for embedded systems such as smartphones. Another approach that is less resource demanding is paravirtualization. Unlikely full virtualization where the guest OS is not aware of running in a virtualised environment, in paravirtualization it is necessary to modify the guest OS to boost performance. Paravirtualization for smartphones is currently under development and several solutions exist (e.g.,Trango, VirtualLogix, L4 microkernel, L4Android).

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.

- All the virtualization solutions suffer from having a coarse grained approach(i.e., the virtualisedenvironments are completely separated, even when this might be a limitation for interaction).
- Other limitation is the hardcoding of the environment specification. Environments cannot be defined by the user/company according to their needs but they are predefined and hardcoded in the virtual machine.
- Furthermore, the switching among environments always require user interactions and it could take a significant amount of time and power. While researchers are improving some of these aspects, the complete separation of virtual machines and the impossibility to change or adapt their specifications remain an open issue.

WORLDWIDE smartphone sales totalled 250 million units in the third quarter of 2013, up 46 percent from the same quarter of 2012 [1]. In the smartphone domain, the Android OS is by far the most popular platform with 82 percent market share. Those figures clearly show the pervasiveness of Android, mostly justified by its openness to third party developers. Smartphones allow end users to perform several tasks while being on the move. As a consequence, end users require their personal smartphones to be connected to their work IT infrastructure. More and more companies nowadays provide mobile versions of their desktop applications. Studies have shown that allowing access to enterprise services with smartphones increases employee productivity [2]. An increasing number of companies are even embracing the BYOD: Bring Your Own Device policy [3], leveraging the employee's smartphone to provide mobile access to company's applications. Several device manufacturers are even following this trend by producing smartphones able to handle two subscriber identification modules (SIMs) at the same time. Despite this positive scenario, since users can install third-party applications on their smartphones, several security concerns may arise. For instance, malicious

Focusing on security, Android combines two levels of enforcement [11], [12]: at the Linux kernel level and the application framework level. At the Linux kernel level Android is a multi-process system. During installation, an application is assigned with a unique Linux user identifier (UID) and a group identifier (GID). Thus, in the Android OS each application is executed as a different user process within its own isolated address space. All files in the memory of a device are also subject to Linux access control. On a

Linux, file access permissions are set for three types of users: the owner of the file, the users who are in the same group with the owner of the file and all other users. For each type a tuple of read, write and execute (r-w-x) permissions is assigned. In Android, by default, the files in the user's home directory can be read, written and executed by the owner and the users from the same group as the owner. All other users cannot work with these files. So as different applications by default have different user identifiers files created by one application cannot be accessed by another. At the application framework level, Android provides access control through the inter-component communication (ICC) reference monitor. The reference monitor provides mandatory access control (MAC) enforcement on how applications access the components. In the simplest form, protected features are assigned with unique security labels—permissions. Protected features may include protected application components and system services (e.g., Bluetooth). To make the use of protected features, the developer of an application must declare the required permissions in its package manifest file: AndroidManifest.xml. As an example, consider an application that needs to monitor incoming SMS messages, Android Manifest. xml included in the application's package would specify: . Permissions declared in the package manifest are granted at the installation time and cannot be modified later. Each permission definition specifies a protection level which can be: normal (automatically granted), dangerous (requires user confirmation), signature (requesting application must be signed with the same key as the application declaring the permission), or signature or system (granted to packages signed with the system key or located in the system image).

CONTRIBUTIONS

This paper presents MOSES, a solution for separating modes of use in smartphones. MOSES implements soft virtualization through controlled software isolation. With MOSES:

- Each security profile (SP) can be associated to one or more contexts that determine when the profile become active. Contexts are defined in term of low level features (e.g., time and location) and high level features (reputation, trust level, etc.).
- Both contexts and profiles can be easily and dynamically specified by end users. MOSES provides a GUI for this purpose. Profiles can be fine-grained to the level of single object (e.g., file, SMS) and single application.
- Switching between security profiles can require user interaction or be automatic, efficient, and transparent to the user.

We implemented MOSES and ran a thorough set of experiments to evaluate its efficiency and effectiveness. The experiments show the feasibility and accepted performance of our solution for storage and energy consumption.

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 virtual environment, where in paravirtualization [4] the guest OS is modified to increase the performance. In smart phones 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 environment 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

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

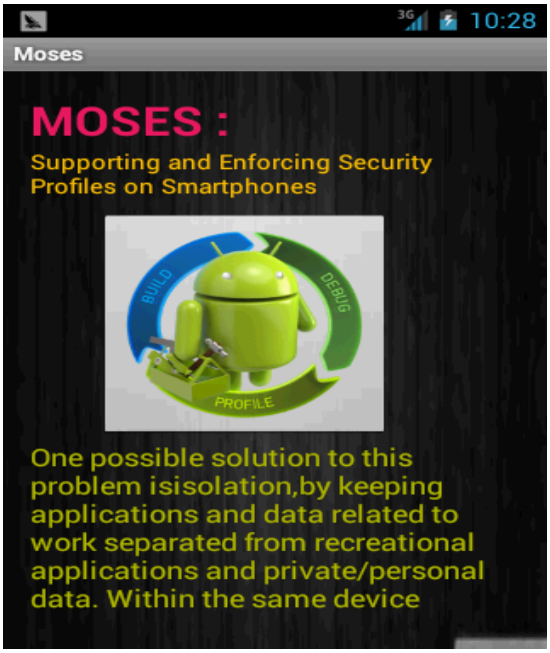
ANDROID SECURITY

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 application components communicate 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

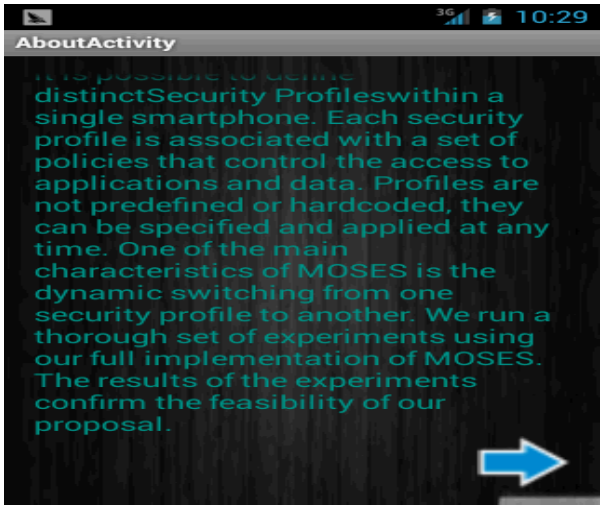
security-savvy developers to easily work and rely on them. The attackers attempts to attack on users applications by such as social engineering attack where in this they try to convince users to install malware and they can also attack to third-party applications. In this paper we discussed the security features of Android platform of specific applications, which are those related to the SMS or browser application.

IMPLEMENTATION

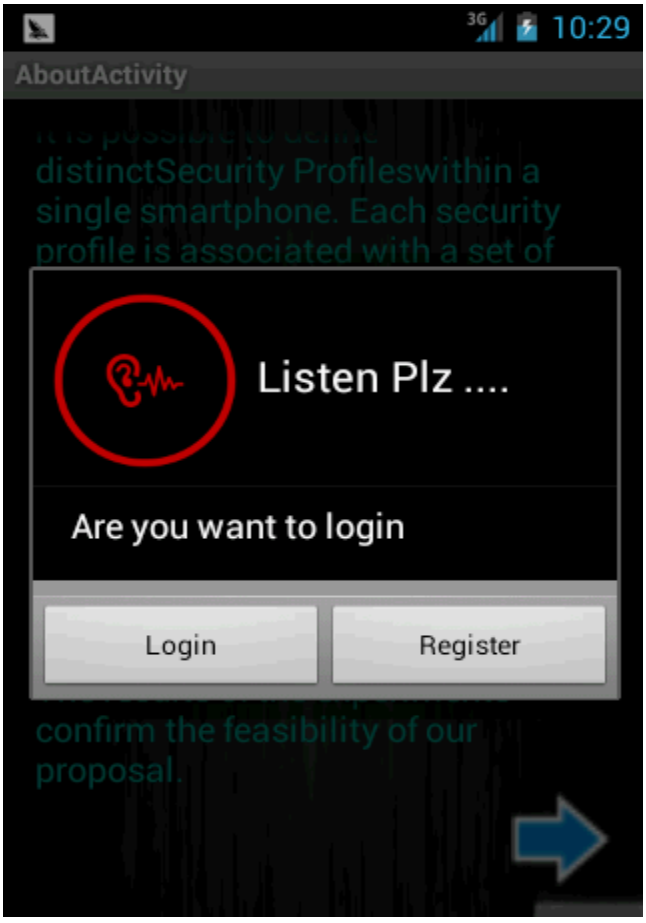
In this section implementation of some key aspects of MOSES are described. Android Open Source Project (AOSP) is used to define the versions in MOSES version 2.3.4_r1.



Main Page

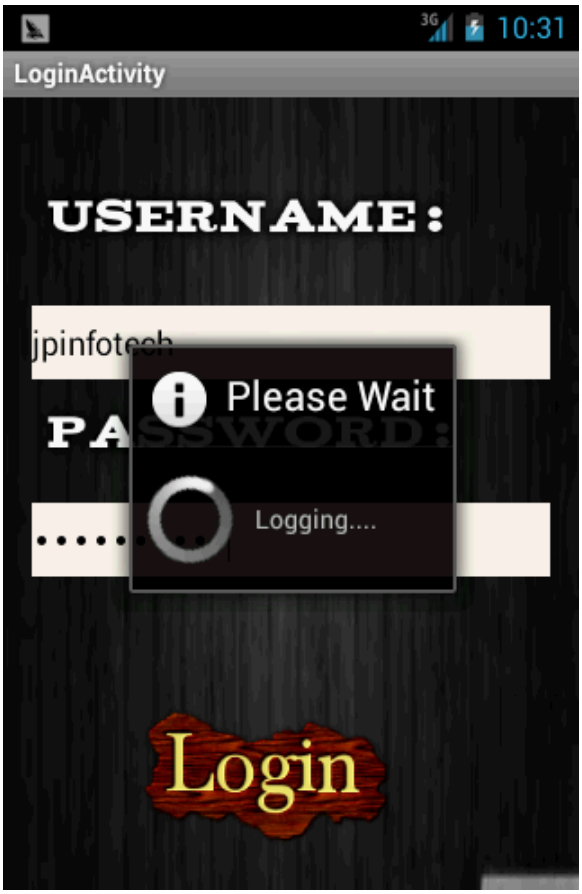


Activity page

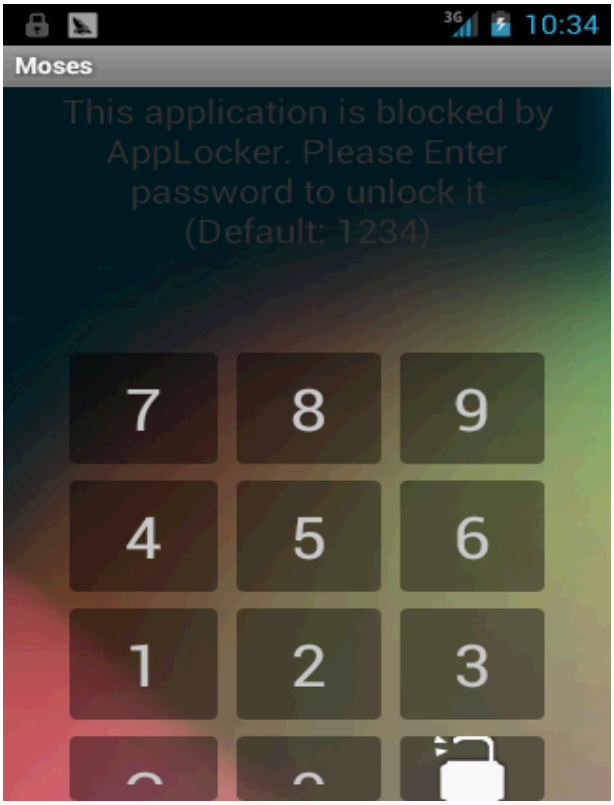


Registration page

ISSN: 2250-3676
Volume-7,Issue-1, 001-008



Login page



Pattern for Moses page

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.

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.

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.

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.and roid.com/ en//devices/tech/ security /reports/Google_Android_Security_2014_Report_Fin al.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-owndevice, 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