Context Related Policies on Data Isolation for Organizational Data using Profiles

[1]Vignesh M, [2]Aishwarya K
[1]M.E Student, [2]Assistant Professor,
[1, 2] Department of Computer Science and Engineering,
Arulmigu Meenakshi Amman College of Engineering, vadamavandal-604410
Kancheepuram, Tamilnadu, India

***Abstract* — General studies have shown that allowing access to enterprise services with smartphones increases employee productivity. Despite this positive scenario, users can install third-party applications on their smartphones, which may provide some malicious applications to access their company confidential information. This poses serious security concerns to sensitive corporate data. One possible solution to this problem is isolation, by keeping applications and data related to work separated from recreational applications. MOSES a solution for separating modes of use in smartphones. MOSES implements soft virtualization through controlled software. 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. Isolation Using MOSES creating profiles for securing confidential data but that the applications that have root access to the system can bypass MOSES protection. A Novel Attribute Based Access Control Enabled MOSES method proposed to overcome the problem of these root application bypassing the security profile protections**.

***Keywords* — MOSES, ABAC, GUI, SP, BYOD, Context**

I. INTRODUCTION

Smartphones allow end users to perform several tasks while being on the move [7]. As a consequence, end users require their personal smartphones to be connected to their work in IT infrastructure [9]. More of the companies nowadays provide mobile versions of their desktop applications [27]. Studies have shown that allowing access to enterprise services with smartphones increases employee productivity [23][1].

An increasing number of companies are even embracing the BYOD: Bring Your Own Device policy [2], leveraging the employee's smartphone to provide mobile access to company's applications. Since users can install third-party Applications on their smartphones, several security concerns may arise. Malicious applications may access emails, SMS and MMS stored in the smartphone containing company confidential data [4]. Even more worrying is the number of legitimate applications harvesting and leaking data that are not strictly necessary for the functions the applications advertise to users. This poses serious security concerns to sensitive corporate data, especially when the standard security mechanisms offered by the platform are not sufficient to protect the users from such attacks. Keeping applications and data related to work separated from recreational applications and private/personal data [13]. Within the same device, separate security environments might exist. As long as applications from the second environment are not able to access data of the first environment the risk of leakage of sensitive information can be greatly reduced.

Virtualization technologies where different instances of an OS can run separately on the same device [18]. It is still too resource demanding for embedded systems such as smartphones. Unlikely full virtualization where the guest OS is not aware of running in a virtualized 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. Furthermore, the switching among environments always require user interactions and it could take a significant amount of time and power [5]. User populations of information systems have explained, the challenge of controlling access to resources using security policies has grown. Researchers and system developers have simplified the administrative process by using groups of users who have the same authorizations. User groups were the precursor to role-based access control [16].

RBAC group's permission into roles and requires all access to occur through the RBAC system. Groups of permission can then be readily provided to users in the simple operation of assigning roles. An enterprise's role must be engineered to support security and business rules. Overtime, enterprises recognized a need for going beyond RBAC's groups of users and permission. They needed to include attributes, such as time of day and user location, for distributed. Dynamically changing systems. This period, attribute-based access control was identified as a replacement for or identified as a replacement for or adjunct to RBAC.

ABAC uses labeled objects and user attributes instead of permission to provide to provide access control in a flexible manner. It was argued that ABAC could provide the flexibility needed in access control and that, if desired, RBAC could coexist with ABAC simply by considering a role as another attribute. Because ABAC doesn't user roles with permission, it also avoids the need to engineer those roles and permissions. RBAC researchers have come up with several schemes or providing this attribute component using constrained role. RBAC the permission granted to a user through roles must be evaluated to determine

if the desired access will be granted. That is, a user is pre-assigned a set of roles with RBAC; While ABAC permissions can be acquired dynamically by virtue of the user's attributes. RBAC permissions are defined as an operation on an object, so only defined combinations of operations and objects are allowed .to achieve this granularity of access in ABAC requires rule sets that apply when attributes are evaluated.

ABAC is newer, simpler to implement, and accommodates real-time environmental states as access control parameters.

II. ANDROID SECURITY

Google Android is a Linux-based mobile platform developed by the Open Handset Alliance (OHA) [8]. Most of the Android applications are programmed in Java and compiled into a custom byte-code that is run by the Dalvik virtual machine (DVM). Android applications are built combining any of the following four basic components. Activities represent a user interface; Services execute background processes; Broadcast Receivers are mailboxes for communications within components of the same application or belonging to different apps; Content Providers store and share application's data.

Application components communicate through messages called Intents. Focusing on security, Android combines two levels of enforcement [7], [8]: at the Linux kernel level and the application framework level. At the Linux kernel level Android is a multi-process system [24]. During installation, an application is assigned with a unique Linux user identifier (UID) and a group identifier (GID). 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 [25]. So as different applications by default have different user identifiers files created by one application cannot be accessed by another. In the simplest form, protected features are assigned with unique security labels—permissions. Protected features may include protected application components and system services to make the use of protected features, the developer of an application must declare the required permissions in its package manifest file:AndroidManifest.xml [11].

III. RELATEDWORK

Android Security Extensions

In Android, at installation time users grant applications the permissions requested in the manifest file. Supports an all-or-nothing approach, meaning that the user has to either grant all the permissions specified in the manifest or abort the installation of the application. Moreover, permission cannot be revoked at runtime.

Crepe allows a user to create policies that can automatically control the granting of permissions during runtime [9].

Taintdroid proposes dynamic taint analysis to control how data flow between applications [3]. In Taintdroid, taints are statically associated with predefined data sources, such as the contact book, SMS messages, the device identifier (IMEI), etc. Taintdroid tracks the flow of tainted data and notifies the user if the tainted data leave the device through the outbound network connections. By using Taintdroid's tainting capability, the mandatory access control implemented in [11], [12] considerably diminishes the effect of root exploits.

Bring Your Own Device Approaches

AppGuard system for instance, is a standalone Java application that disassembles apk file, inline security checks before dangerous instructions according to a selected policy and then reassembles and signs the package [12]. Thus at runtime, before executing a dangerous instruction AppGuard performs a security check and if the instruction is not allowed according to the policy an exception is thrown.

Virtualization

Virtualization provides environments that are isolated from each other, and that are indistinguishable from the "bare" hardware, from the OS point of view [17].

Full virtualization:

Unmodified guest OS can be run on the virtual machine directly. The guest OS does not realize that it is running in a virtualized environment. Full virtualization is usually realized either by Dynamic Binary Translation (DBT) (e.g. VMware ESX [17] [18]) or by hardware assistant (e.g. Xen [19]).

Para-virtualization:

Unlike full virtualization approach, modifications are needed to de-privilege the guest OS. Usually modifications of system call interface, memory management, and interrupt handling are necessary. The advantage of para-virtualization approach is its high performance. Examples of this approach are Xen and L4 microkernel based virtualization approach (L4Linux [6]).

The requirements for virtualization on mobile phones are quite different from virtualization on high performance systems [18]. Some suitable virtualization technologies for high performance systems may become unapplicable on mobile phones due to hardware resources and power consumption limitations [20]. For example, in full virtualization by DBT, the executed instructions are intercepted and replaced in real time. This is computationally intensive and unsuitable for mobile systems. Virtue, Para-

virtualization is currently the emerging solution for virtualization in mobile phones. Examples are Trango (current VMWare MVP) [17], VirtualLogix virtualization technology [20] and L4 microkernel virtualization technology [6]. This is substantially determined by the fact that on mobile phones, high performance efficiency is preferred because of limited resources. Among all these available embedded system virtualization solutions, only the L4 microkernel virtualization approach is open-source, which is a big advantage in research work. It gives us the possibility to deeply understand the embedded system virtualization approach and allows us to do detailed analysis and evaluation and the hardware's dedicated for virtualization, which are usually available for PC and server systems are not yet available at the embedded market. With its high performance. However, there are still several barriers for the adoption of virtualization in mobile devices. The main one is that ARM architecture, which is the most popular architecture for mobile devices, has a non-virtualizable instruction set so as efficiency is a major concern in embedded virtualization, full virtualization approaches (emulation and binary translation) are not yet applicable for these devices because they are computational expensive [21]. Thus, for embedded devices paravirtualization is used, which requires source-code modification of guest operating system [20].

Other Approaches

Besides the above approaches there few other solutions. Modified Android framework to supportdual mode of operation, private and enterprise [26]. The modification allows to restrict the use of communication capabilities of a phone, to force communication through enterprise.

VPN and have an encrypted external storage in enterprisemode. The authors of Trust Droid proposed to monitorIPC communications, network traffic and file system accessto separate data exchange between different domains, forinstance, between enterprise and personal environments [28].

Furthermore, all of them offer only profiles predefined by the solution developers.

IV. ABAC OVERVIEW

This section provides an overview ofapproach named Attribute Based Access Control (ABAC).

ACLs and RBAC are in some ways special cases of ABAC in terms of the attributes used. ACLs work on the attribute of "identity". RBAC works on the attribute of "role". The key difference with ABAC is the concept of policies that express a complex Boolean rule set that can evaluate many different attributes. While it is possible to achieve ABAC objectives using ACLs or RBAC, demonstrating AC requirements compliance is difficult and costly due to the level of abstraction required between the AC requirements and the ACL or RBAC model. Another problem with ACL or RBAC models is that if the AC requirement is changed, it may be difficult to identify all the places where the ACL or RBAC implementation needs to be updated. Instead, when a subject requests access, the ABAC engine can make an access control decision based on the assigned attributes of the requester, the assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions. Under this arrangement policies can be created and managed without direct reference to potentially numerous users and objects, and users and objects can be provisioned without reference to policy. An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

Attributes are characteristics of the subject, object, or environment conditions. Attributes contain information given by a name-value pair.

Subject is a human user, such as a device that issues access requests to perform operations on objects. Subjects are assigned one or more attributes. For the purpose of this document, assume that subject and user are synonymous.

An object is a system resource for which access is managed by the ABAC system, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information. It can be the resource or requested entity, as well as anything upon which an operation may be performed by a subject including data, applications, services, devices, and networks.

An operation is the execution of a function at the request of a subject upon an object. Operations include read, write, edit, delete, copy, execute, and modify.

Policy is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the subject, object, and possibly environment conditions.

Environment conditions: operational or situational context in which access requests occur. Environment conditions are detectable environmental characteristics. Environment characteristics are independent of subject or object, and may include the current time, day of the week, location of a user, or the current threat level.

As new users arrive, old users leave, and characteristics of subjects change, these subject attributes may need to be updated. Every object within the system must have at least one policy that defines the access rules for the allowable subjects, operations, and environment conditions to the object. This policy is normally derived from documented or procedural rules that describe the business processes and allowable actions within the Organization. For example, in a hospital setting, a rule may state that only authorized medical personnel shall be able to access a patient's medical

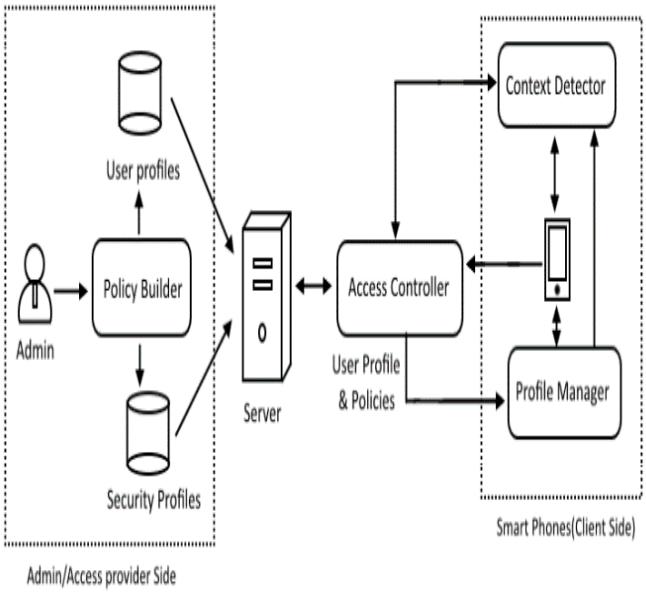


Fig: Architecture Diagram

Record. In some system, if the object is a document with a Record Type Attribute of Patient Medical Record, then the Medical Record Rule will be selected and processed so that the subject with a Personnel Type Attribute value of Non-Medical Support Staff trying to perform the Read operation will be denied access and the operation will be disallowed. Note that this is only one approach to implementing the connection between attributes and rules.

V. ARCHITECTURE

ABAC consists of components presented in Fig 1 Central to ABAC is notion of the context. The component Context Detector System is responsible for detecting context activation/deactivation. When such an event happens, the Context Detector System sends a notification about this to the Security Profile Manager.

Profile Manager holds the information a SP with one or more Contexts. The profile Manager is responsible for the activation and deactivation of SPs. The Moses Policy Manager acts as the policy administrator point (PAP) in MOSES. It provides the API for creating, updating and deleting MOSES policies. It also allows a user to define, modify, remove monitored Contexts and assign them to SPs.

Access control handler module is used by the access provider to apply different the security policies to different access rights [10]. It provides login functionality to users and fetches their attributes values and assigns the equivalent access rights. This module also verifies whether the user is accessing the right content according to their assigned access right by checking the each and every event of the user from the background.

Access control handle model consists of several components for handling profile management and it created different users ad their different access right. it also provides login functionality to users and fetches their attributes values and assigns the equivalent access rights. It display the added user. it also verifies the user is accessing the right content according to their assigned access right by checking the each and every event of the user from the background. The user access rights are maintained by access control handler who provides particulars rights to user according to their role. As per the role each and every user has general rights which will be maintained in general policies. From the user perspective information share offers a complete access control panel allowing users to access the system and authorized objects according to access control policies active during their sessions.

The cases where all the users are present or all the roles are present are not desirable. Having all the users on all the system violates the latest privilege principle. If these users have no access to resources on that system should not have an account the system that enables them to login. Similarly having all roles in all the system is not appropriate as we do not wish to have roles that are not relevant to a server. For example, an engineer role may not be needed in a sales database .in this work assume the case that every system does not have all the roles and all the users.by addressing this general case we clearly cover all the other cases also.

Context Detector uses a core event specification language for construction of emergency policies [22]. Emergencies are detected by continuously monitoring the data stream and if there are some constraints on those streams the emergencies are triggered to correlate emergency events with database queries, as they are triggered only after setting constraints so detecting the constraints is likely querying the datastreams for particular output. There are several languages for specifying the event patterns. There is no standard language for specifying the event being followed yet. Emergency event is detected when a particular event pattern is determined in the stream .Whenever emergency event is detected in the data stream, an instance is created and forwarded to the end user. Emergency depends on emergency events like init, end and timeout.

Whenever emergency is detected it means init event has occurred, if the emergency ends then end event occurs, if the event goes in an infinite loop then after a particular timeout it stops. Emergency policy requires emergency managers to define the access control policies which will be used during emergencies to grant the access to the user. Such policies should also be able to detect emergencies which can activate access control policies should be enforced by the system and separate instance of the template should be created.

The polices should follow the specification enforced by emergency managers and it should accomplish the set of desired actions after giving the access to the users. The advantage of such policy is that based on the events detection, access control policies can be activated or deactivated.

Profile manager holds the information linking a security profiles with one or more context. The Security Profile Manager is responsible for the activation and deactivation of security policies.

It acts as a policy decision point (PDP) in our system which holds rules manager (i.e. read, write, update, edit, etc.,) and the security templates.

Policy builder which is handled by the Administrator form the corporate network side. The Policy Manager acts as the policy administrator point (PAP) in the project. Administrator uses this module to create users/employees and their profiles. It also provides the API for creating, updating and deleting policies. It also allows a user to define, modify, remove monitored Contexts and assign them to security policies. The enforcement of separated SPs require special components to manage application processes and file system views.

When a new SP is activated, it might deny the execution of some applications allowed in the previous profile. If these applications are running during the profile switch, then it need to stop their processes.

To separate data between profiles different file system view are supported.

VI. IMPLEMENTATION

Access control is the mechanism by which services know whether to honor or deny requests.

Identification: Assigning a responsible party for actions. A responsible party may be a person or a non-person entity (NPE), such as a computer or a router.
Authentication: The means used to prove the right to use an identity, take on a role or prove possession of one or more attributes.
Authorization: The means of expressing the access policy by explicitly granting a right.
Access Decision: Using some combination of the other three to decide whether or not a request should be honored.

In proposed system, a generic Access control Mechanisms used for securing the Corporate Data on a Smart Phone Device [27]. Which Provides an Isolation environment for corporate data and also restrict the access of malware apps bypassing the mobile phone security to steal the corporate data. Also Enable the Authentication and authorization Mechanisms.

ACLs and RBAC are in some ways special cases of ABAC in terms of the attributes used. ACLs work on the attribute of "identity". RBAC works on the attribute of "role". The key difference with ABAC is the concept of policies that express a complex Boolean rule set that can evaluate many different attributes. While it is possible to achieve ABAC objectives using ACLs or RBAC, demonstrating AC. Requirements compliance is difficult and costly due to the level of abstraction required between the AC requirements and the ACL or RBAC model. Another problem with ACL or RBAC models is that if the AC requirement is changed, it may be difficult to identify all the places where the ACL or RBAC implementation needs to be updated.

Instead, when a subject requests access, the ABAC engine can make an access control decision based on the assigned attributes of the requester, the assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

Under this arrangement policies can be created and managed without direct reference to potentially numerous users and objects, and users and objects can be provisioned without reference to policy. An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

Attributes are characteristics of the subject, object, or environment conditions. Attributes contain information given by a name-value pair.

Subject is a human user, such as a device that issues access requests to perform operations on objects. Subjects are assigned one or more attributes. For the purpose of this document, assume that subject and user are synonymous.

An object is a system resource for which access is managed by the ABAC system, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information.

It can be the resource or requested entity, as well as anything upon which an operation may be performed by a subject including data, applications, services, devices, and networks.

An operation is the execution of a function at the request of a subject upon an object. Operations include read, write, edit, delete, copy, execute, and modify.
Policy is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the subject, object, and possibly environment conditions.
Environment conditions*:* operational or situational context in which access requests occur. Environment conditions are detectable environmental characteristics. Environment characteristics are independent of subject or object, and may include the current time, day of the week, location of a user, or the current threat level.

As new users arrive, old users leave, and characteristics of subjects change, these subject attributes may need to be updated. Every object within the system must have at least one policy that defines the access rules for the allowable subjects, operations, and environment conditions to the object. This policy is normally derived from documented or procedural rules that describe the business processes and allowable actions within the Organization. For example, in a hospital setting, a rule may state that only authorized medical personnel shall be able to access a patient's medical record. In some system, if the object is a document with a Record Type Attribute of Patient Medical Record, then the Medical Record Rule will be selected and processed so that the subject with a Personnel Type Attribute value of Non-Medical Support Staff trying to perform the Read operation will be denied access and the operation will be disallowed. Note that this is only one approach to implementing the connection between attributes and rules.

The entire concepts were implemented in the Android emulator. Easy tutorials for creating apps inside the emulator if doesn't have smartphone. These are the initial requirements for to deploy the android into a personal computer they are following as Eclipse, Android SDK, Google Maps and Android emulator (i.e. virtual machine).
Context Detection can automatically switch SPs based on the current context. The Context Detector System is responsible for monitoring Context definitions and for notifying the listeners about the activation or deactivation of a context. The Profile manager component, which is one of these listeners, is notified about the change through the callback functions (context_id) which corresponds to activation and deactivation of a context respectively.so as ABAC Enabled MOSES context detection functionality is decoupled from the rest of the system, it may be easily extended by integrating other context detection solutions. When the emulator starts up, MOSES selects form the database information about all contexts and corresponding SPs.

File System Virtualization

Smart phone file system structure is quite stable,i.e the system forces an application to store its files in the application's "home" directory .that is /data/data/<package name>/(<package name> is the package name of the application).during the installation of an application; Smartphone OS creates this "home "folder and assigns it OS file permission to allow only the owner of the directory to access the data stored in it.To provide applications with different data depending on a currently running SPs, Polyinstantiation of "data" folder may be used.i.e for each SP a separate name space ,which points to different "physical" data folder depending on the identifier of a SP,may be created. In ABAC enabled MOSES the described approach is used with two modifications. The first let the system to store all "physical" data directories under some parent directory (/data/ABAC-MOSES/private).

The second modification creates the bindings not between the whole data folder and its "Physical "counterpart, but for separate application folder.

The Content Storage section id responsible providing the functionality of management of application as well as corporate data. In particular, it receives the list of applications package names that are allowed to execute in a SP.If a new SP is created, these initial data are copied to the physical directory providing the application of a fresh copy of its initial data as if the application has just been installed. If this process finishes successfully, the content storage stores the name of this list of application along with the attributes (ABAC).Thus, the process of polyinstantiation is completely transparent for the applications: after the storage of applications work with the same paths as usual, although these paths point to another "phycial" location. Thus there is no need to modify the application to support the separation of data between different SPs.

Dynamic Application Activation

Each SP is assigned with a list of application UIDs that are allowed to be run when this profile is active. Each application during the installation receives its own UID.ABAC enabled MOSES uses these identifiers to control which applications can be activated for each SP.It should be mentioned that some packages can share the same UID.this happens if the developer of these applications have explicitly assigned the same value to sharedUserId property in the manifest files of the applications, and signed these packages with the same certificate. Thus, during the installation of these applications, the smartphone OS system distinguish these applications and if one of them is allowed in one profile the other will be allowed as well.

During the SP switching, the Profile manager selects from ABAC enabled MOSES database the list of UIDs, which are allowed in the activated profile, and stores it into the set of allowed IDs. To control the launch of applications services and activities, the hooks into the retrieveServiceLocked and startActivityMaywait methods of the ActivityManagerService and ActivityStack classes correspondingly are put. These hooks communicate with the ProfileManager and check against the set of allowed apps if a component of an application can be launched.Additonally controls the appearance of application. Additionally, the ProfileManager controls the appearance of an application icons in Smartphone's Launcher application. When a new SP is activated, only the icons of the allowed applications for this profile will be displayed.

Attribute Based Access Control Policies

Within each SP, ABAC enabled MOSES enforces an attribute based access control model. The idea is that within each SP, users can define fine-grained access control policies to constraint application behavior. For instance, the user may want to

deny an application to read the file on an external storage.in this case, the user may write a policy which will still let the application to run within the profile but the access of this application to files on an external storage will be limited.
Subjects represents the application to which the rule is applied. The application UID is retrieved through the GUI provided by ABAC enabled MOSES for management listing the applications installed in the system.

Operation is the action that the subject is executing. The value of this attributes is dependent on the control hooks added by MOSES in the smartphone framework. Each hook communicate with a special class in the framework library that processes the information obtained from the operation hook. For instance, for controlling access to content provider, the injected hooks in the content resolver class. Similarly, for network and file system operation have injected in the core library. ABAC enabled MOSES supports the following Operations.
Content Provider: Query Content Provider, insert in content Provider, and update Content Provider, Delete Content Provider.
Location Provider: Get Last Known Location, Request Location Updates, ADD Proximity Alert, and Requested single Updated.
Network: receive Internet Data, Send Data to the Internet.
File System: Read from a File, Write to a File.
Device ID: Get Device ID.

SP-name attribute represents the SP name where the policy is valid. The decision that can be assigned to Policy rules are: Allow, Deny, Update,Edit, Allow with perform, read, and write of the first two are obvious. The decision Allow_With_Perform corresponds to allow with a restrictive obligation that performs additional actions of the data returned by the operation. It is possible that two or more rules may be defined for the same attribute values. To resolve these conflicts, the user should assign a priority value to each rule. In this case, the decision of the rule with the highest priority will have precedence over the decisions of other rules; in the case of equal priorities, then the last inserted rule takes priority.

For some combinations of attributes values, it might be the case when no rules apply. In this case, our system uses a default value (either allow or deny), which is assigned to the SP.

Profile Management

To give a user the ability to manage the SPs in her device. The MOSES PolicyGUI application is developed. This is a system application signed with a system key and assigned with a special permission. This allows MOSES PolicyGUI application to communicate with the MOSESPOLICYMANAGER and manage the SPs.Due to lack of space, we will not show screenshots provides. The MOSES PolicyGUI manages Contexts and SPs.we develop an application that allows a user to easily configure MOSES functionality. To define a new SP the application provides a wizard that guides the user through the steps. Fig a) provides a Login Credentials to the respected user of a corporate network. Fig b) shows the relationship between the mobile emulator and as well as the server side authentication it also show that what kind of user and their Team Head, Team Size, Attribute Access Control for IT resources. Fig c) show the search Key scenarios for respected IT resources. Fig d) shows the Corporate Server Administration. Internal control follows as how to assign application to the SP and in the next phase assign an ABAC policy rules to deny or allow the particular user or even some cases users' application as well. Which also manage all the smart phone user details examplelikei.e.Android, IOS, windows, Blackberry. Including the user location details, password management, and their content details.Team details, their Team Leader and the relevant team details. Which also enable the options for searching a person who belong different group. Enable the permission(r-w-e) details for the user profiles. And also enable the dynamic option for adding a new employee to the ABAC database and make the permission attributes for the new arrival user.Enable the categories of documents like doc, ppt, emails, videos something like that which has ABAC properties of Read, Write, Edit, and other related properties of ABAC enabled MOSES. And also enable the options for the user who wants to update their details on their ABAC enabled database.

(a) (b) (c) (d)

Fig: Screenshots of ABAC enabled MOSES :(a) Profile GUI and Sever homepage, (b) Profile Management List, (c) Searching areas (including profile updates, Location update, etc.., (d) Administration side including Mobile and Server.

VII. EVALUATION

In this section, we report on the thorough experiments we ran to evaluate the performance of ABAC enabled MOSES. For all the experiments, we used a GOOGLE Nexus S phone.

Energy Overhead

To measure the energy overhead produced by ABAC enabled MOSES, We performed the following tests. We charged the battery of our device to the 100 percent .then, every 10 minutes we run our application along with the system preinstalled application like screensaver, restore application. Via monkey runner scripts and some other application .each experiments lasted for a total of 120 minutes. We execute this experiment for three types of system: real time (i.e. smart phone) and simulation (virtual environment emulator).During each experiment, every 10 seconds, out service measured the level of the battery and wrote this value into a log file. For each of the considered systems, we executed the test 10 times and averaged the obtained values. The results of this experiment are reported in below diagram.

This show that the fact that ABAC enabled MOSES is just running, or even switching context does not incur a noticeable energy overhead.

Storage Overhead:

One of the most significance overhead produced by is the storage overhead. In fact, the separation of data for different SPs means that some application information will be duplicated is different profiles. One additional copy of application data is required to store initial information of all applications. If a new SP is created, we need a clean copy of application data to be replicated into this new profile. Hence, ABAC enabled MOSES stores a copy of application data just after the installation of the application: this copy is later used for replication when anew a SP is created. it should be mentioned that for MOSES only the initial data of applications are duplicated. The data produced by application during runtime are not replicated between SPs. Second, the data of application, which are not allowed in a profile, are not copied into the profile. When comparing ABAC enabled MOSES with competitor approaches, which produces less storage overhead. Forinstance, incase of mobile virtualization not only application data are duplicated, but also application executable and an operating system. Thus ABAC enabled MOSES adds less overhead comparing to this set of approaches because it only works with one copy of application executable.
Moreover, other improvements are possible for ABAC enabled MOSES, e.g. currently for each SPs MOSES stores its own copy of the shared libraries of an application, instead they could be shared among the different profiles.

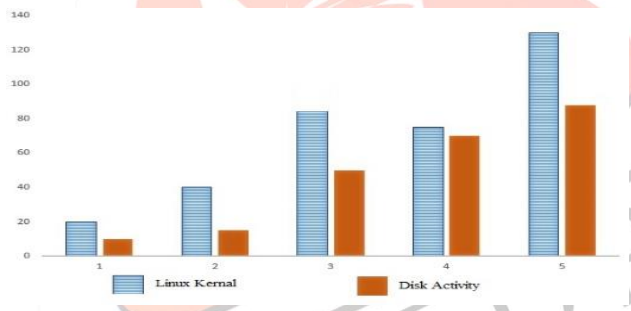


Fig: Systrace

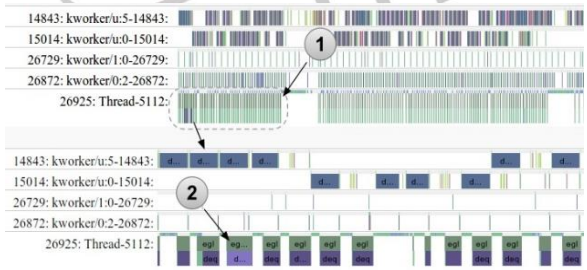


Fig: Process diagram for smart phones

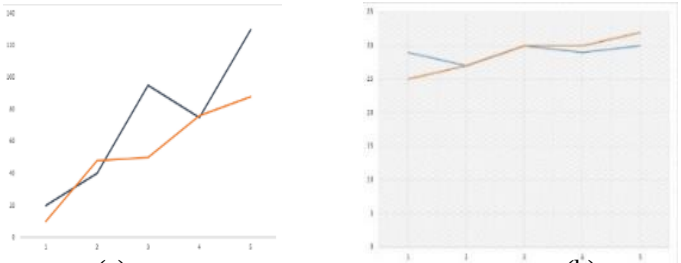
Micro benchmark:

To assess the overall performance of our system, we decided to run a set of experiments with a benchmarking system. In particular, we used the systrace, androidlint tools and device anywhere [14][15]. These benchmark runs a set of test which allows a user to assess different aspects of virtual machine performance. The benchmark does not produce absolute values for the tests .instead it uses internal scorings measures, which are useful only in case of comparison with other systems, which also includes the details of Appgurad and SRT privacy monitoring tool [12] the analysis results are showed below with diagrammatic representation.
Systrace helps to analyze how the execution of an application fits into the larger smartphone environment, letting you see system and application process execution on a common timeline. The tool allows to generate highly detailed, interactive reports form devices running Android version 4.4 and higher, such as described in the emulator section in the implementation part.

In order to create a trace of an application, set up the device for debugging, connect it to the development systems, and install the application. Some types of trace information, specifically disk activity and kernel queues, require that which have root access to the device. However, most systrace log data only requires that the device be enabled for developer debugging. Systrace traces can be run either from a command line or from a graphical user interface [14].

This guide focuses on using the command line options.to user systrace effectively with devices running Android 4.4 and higher, it must configure the types of processes that want to trace before running a trace.

The tool can gather the following types of process information.



(a) (b)

Fig: a) User Application and b) Number of Rules

General system processes such as graphics, audio and input processes and low level system information such as CPU, kernel and disk activity.

To set trace tags for systrace using command-line;

1. User the –set-tags option:

$ cd android-sdk/platform-tools/systrace

$python systrace.py –set-tags=gfx,view,wm

2. Stop and restart the adb shell to enable tracing of these processes.

$ adb shell stop

$adb shell start

To run a trace using the current trace tag settings:

1. Make sure the device is connected through a USB cable and is enabled for debugging.
2. Run the trace with the low-level system trace options and limits you want, for example:

   $ Python systrace.py –cpu-freq—cpu-load—time=10 –o mytracefile.html

3. On the device, execute any user actions you want be included in the trace.

After we analyzed a trace using Systrace, it lists the location of the output file and it can open the report using a web browser. However this section provides some general instructions on how to analyze a trace. The reports generated by Systrace are interactive, allowing to zoom into an out of the process execution details. Use the w key to zoom in, the S key to zoom out, the A key to pan left and the d key to pan right. Select a tasks in timeline using mouse to get more information about the task. A well-behaved application executes many small operations quickly and with a regular rhythm, with individuals operations completing within few milliseconds, depending on the device and the processes being performed.

The trace except in figure 2 shows a well–behaved application with a regular process rhythm (1).the lower section of figures 2 shows a magnified section of the trace indicated by the dotted outline,

which reveals some irregularity in the process execution.in particular, one of the wider task bars, indicatedby (2), is taking slightly longer (14 milliseconds) thanother,

Similar tasks in this thread, which are averaging between 9 and between 12 milliseconds to complete.

This particular task execution length is likely not noticeable to a user, unless it impacts another process with specific timing, such as a screen update. Long running processes show up as thicker than usual execution bars in a trace. These thicker bars than bars can be depicted as application performance. When they show up in trace format. Zoom in on the process using the keyboard navigation. Shortcuts to identify the task causing the problem, and click on the task to get more information .we also see that other processes running at the same time, looking for a thread in one process that is being blocked by another process.

8. CONCLUSION AND FUTURE WORK

An Extension of the Access Control model with the Possibility of defining Administration

policies.i.e which subjects are enabled to define policies and over which scope. A set of correctness have been defined to avoid useless activation/deactivation of emergencies. Performs the authentication / authorization process to the end users side as well as server side too.it can give solution to conflict holding problem.

In future work, the similar approach can be extended with the help of cloud based concept for storing large amounts of access control profiles to overcomes the problem of server side overhead and also has a plan to include some sensor-based mechanisms to achieve authentication and authorization and also RBAC is role-centric and ABAC is attribute-centric. Once roles become attributes, the advantages of RBAC are lost. Role names are still associated with user s, but the consideration that roles are collections of permissions is no longer the case. Attribute-centric and role-centric access control models. Attribute centric access control is where attributes control what resources a user can access. A role name (not a role, since a role has permissions in addition to its name) can be included in the attribute-centric model as one of the attributes as-signed to a user.

Thus, attribute-centric access control doesn't encompass RBAC, because permission aren't included in the model. When combined judiciously, the combination can provide access control that's scalable,flexible,auditable,and understandable.Significantly,current research in this topic includes the Role-Centric Attribute-Based Access Control(RBAC) can be further defined and combining both roles and attributes in a reliable manner that preserves the best features of both access control methods.

REFERENCES

[1] Gartner Says Smartphone Sales Accounted for 55 Percent of Overall Mobile Phone Sales in Third Quarter of 2013,http://www.gartner.com/newsroom/id/2623415,2014.

[2]Using Establishes a Bring Your Own Device (BYOD) Policy, http://www.insecureaboutsecurity.com/ 2011/03/14unisys_establishes_a_bring_your_own_device_byod_policy/,2014.

[3]W.Enck,P.Gilebert,B.G.Chun,L.P.cox,J.Jung,P.McDaniel,andA.N.Sheth,"Taintdroid:An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphones, "Prc.Ninth USENIX Conf.Operating Systems Design and Implementation(OSDI 10),pp.1-6,2010 .

[4]C.Gibler, I.Crusell, I. Erickson, H.Chen. "Android Leaks: Automatically Detecting Potentials Privacy Leaks in Android Applications on a Large Scale, "Proc.Fifth int'l Conf.Trust and trustworthy Computing (Trust'12), pp.291-307, 2012.

[5]Y.Xu, F.Bruns, Gonzalez, S.Trabolusi, K.Mott, A.bilgic,"Performance Evaluation of Para-virtualization on Modern Mobile Phone Platform." Proc.Int'l Conf.Computer, Electrical and Systems Science ad Eng. (ICCESSE '10), 2010.

[6]M.Lange, S.Liebergeld, A.Lackorzynski, A.Warg and M.Peter,"L4AndroidLA Generic Operating System Framework for Secure Smartphones and Mobile Devices (SPSM '11), pp.39-50, 2011.

[7]Android,http://www.andoid.com/2014.

[8]A.Shabtai, Y.Fledel, U.Kanonov, Y.Elivici, S.Dolev and C.Glezer,"Google Android: A Comprehensive Security Assessment,"IEEE Security and Privacy,,vol.8,no.2,pp.35-44.Mar./apr.2010.

[9]M.Conti, B.Crispo, E.Fernandes and Y.Zhauniarovich,"Crepe: A System for Enforcing Fine Grained Context Related Policies on Android," IEEE Trans.Infomation Forensics and Security, vol 7, no.5, pp.1426-1438, Oct.2012.

[10]G. Bai, L. Gu, T. Feng, Y. Guo, and X. Chen, "Context-Aware Usage Control for Android,"Proc. Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '10), pp. 326-343, 2010.

[11] S. Bugiel, S. Heuser, and A.-R. Sadeghi, "Flexible and FineGrained Mandatory Access Control on Android for Diverse Security and Privacy Policies,"Proc. 22nd USENIX Conf. Security (Security '13), 2013.

[12]M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von StypRekowsky, "AppGuard—Real-Time Policy Enforcement for Third-Party Applications," Technical Report A/02/2012, Saarland Univ., 2012.

[13] MOSES Project, http://mosesdroid.org/, 2014.

[14]Systrace,http://developers.android.com/,2014.

[15]Android Lint Tool, http://developers.android.com/,2014

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

[17]K. Barr, P. Bungale, S. Deasy, V. Gyuris, P. Hung, C. Newell, H. Tuch, and B. Zoppis, "The VMware Mobile Virtualization Platform: Is That a Hypervisor in Your Pocket?"ACM SIGOPS Operating Systems Rev., vol. 44, no. 4, pp. 124-135, Dec. 2010.

[18] G. Heiser, "Virtualization for Embedded Systems," technical report, Open Kernel Labs, Inc., http://www.ok-labs.com/_assets/image_library/virtualization-forembedded-systems1983.pdf, 2007.

[19]J.-Y. Hwang, S.-B. Suh, S.-K. Heo, C.-J. Park, J.-M. Ryu, S.-Y. Park, and C.-R. Kim, "Xen on ARM: System Virtualization Using Xen Hypervisor for ARM-Based Secure Mobile Phones,"Proc. IEEE Fifth Consumer Comm. and Networking Conf. (CCNC '08), pp. 257-261, 2008.

[20]MobileVirtualization,http://www.redbend.com/en/productsservices/mobile-virtualization, 2014.

[21]J. Andrus, C. Dall, A.V. Hof, O. Laadan, and J. Nieh, "Cells: A Virtual Mobile Smartphone Architecture,"Proc. 23rd ACM Symp. Operating Systems Principles (SOSP '11), pp. 173-187, 2011.

[22]D. Kramer, A. Kocurova, S. Oussena, T. Clark, and P. Komisarczuk, "An Extensible, Self Contained, Layered Approach to Context Acquisition," Proc. Third Int'l Workshop Middleware forPervasive Mobile and Embedded Computing (M-MPAC '11), pp. 6:1-6:7, 2011.

[23]Ubuntu Manuals - PAM Namespaces, http://manpages.ubuntu.com/manpages/maverick/man8/pam_namespace.8.html, 2014.

[24]Mount(2),LinuxManPage,http://linux.die.net/man/2/mount, 2014.

[25]S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, "Practical and Lightweight Domain Isolation on Android,"Proc. First ACM Workshop Security and Privacy in Smartphones and Mobile Devices (SPSM '11), pp. 51-62, 2011.

[26]Divide Webpage, http://www.divide.com/, 2014.

[27]FixmoSafeZone: Corporate Data Protection, http://fixmo.com/products/safezone, 2014.
[28]Trustdroid,http://www.trustdroid.com/,2014.

VIGNESH. M *Received the B.E (Computer Science and Engineering) degree from Panimalar Engineering College, Chennai, and Tamilnadu, India in 2012.Worked as PHP developer for one year. Currently doing his final year degree in M.E (Computer Science and engineering) from Arulmigu Meenaksi Amman College of Engineering, Kancheepuram, and Tamilnadu, India. His research interests include Smartphone Security, Cloud Computing, Big data, Web Services, Smartphone application security, mobile malware, mobile security and privacy.*

K.AISHWARYA *received degree M.E from Anna University Chennai in 2011 and joined as an Assistant Professor in various engineering colleges in Tamil Nadu affiliated to Anna University and has four year teaching experience. She has a membership in computer society of India. She has published international journal. Her current primary areas of research is Network Security.*