iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Moses: Supporting and Enforcing Safety Environments in Predefined Context on Smartphone

Roshni Sherkar[#1], Prof. R. V. Mante[#2], Dr. P. N. Chatur[#3]
[#123]Department of Computer Science and Engineering, Government College of Engineering Amravati

Abstract— Now-a-days smartphone is becoming one of the basic need because of the several exclusive features that it is providing in the form various application, storage, computational power. As a result of this, many corporate firms provides mobile version of their desktop application to their employee in order to be updated always. As smartphone providing such features to end-user there are some security concerns that need to be handled. This manuscript deals with the design and implementation of the SecurePhoneApp application which run on the Smartphone in association with MOSES application. Here, Moses provides internal security to the data on the smartphone whereas SecurePhoneApp prevents the data from external threat.

Keywords— isolation, security profile, virtualization, context, smartphone

I. INTRODUCTION

In today's era the usage of smartphone is keep on increasing day by day. In the smartphone domain, Android OS is gaining more popularity because of its openness to the third-party developer [1] [2]. Smartphones allow user to perform several task while being on the move and remains updated by that time. As a result, many corporate firms are willing to support the usage of the smartphone in order to increase the productivity of business users. Despite this positive scenario, several security concern may arises regarding the data present on the device. As many companies supports the use of smartphone, it is very essential to provide security. I may happen that the third-party application unknowingly tampers the user data. One possible solution to this problem is isolation, by keeping applications and data related to work separated from recreational applications and private/personal data. Within the same device, separate security environments might exist: one security environment could be only restricted to sensitive/corporate data and trusted applications; a second security environment could be used for entertainment where third-party games and popular applications could be installed. 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. This can be done by means of virtualization technique [3]. This manuscript details the implementation of two applications that runs on Android smartphone and allows the user to save their data internally and externally. Two applications mainly are MOSESApp and SecurePhoneApp. We have slightly changed the implementation of MOSES from its original work due to some limitation which are given in the following sections. We have run several experiments to know the compatibility of both the applications which is discussed in the experimental analysis section.

II. OVERALL IMPLEMENTATION OF PROPOSED WORK

By going through MOSES architecture we came to know that it provides internal security to the data present on smartphone but the question arises when we talk about theft concept. Suppose a person is working in some IT sector and using MOSES architecture on his/her smartphone and all over sudden someone has stolen his smartphone at this moment what that person should do because he will lost his important data plus the handset. Even though the handset is not important but what about the data. In order to solve this problem, certain solutions are there so that user can get back the data which very important now-a-days as compared to the handset. Solution is in the form of following diagram.

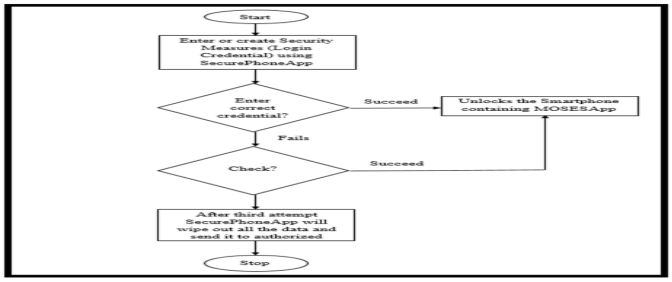


Fig. 1 Flowchart of Proposed Work

©IJRASET 2015: All Rights are Reserved

III.MOSES FRAMEWORK

Moses provides abstraction in order to separate the apps and data related to different context installed on the single device. For example, apps and data related to corporate world are separated from the recreational apps and data. In short, MOSES provides separate compartments to apps and its related data. Here, compartments are called as environment or security profile (SP). In general, SP are the set of policies or rules that regulates which app has to be executed and what data need to be accessed.

A. Architecture

Main part of MOSES is the phenomenon of Context. The ContextDetectorSystem is responsible for activating/deactivating of context. When it happens, the component ContextDetectorSystem notifies about this to the SecurityProfileManager. The SecurityProfileManager handle this information linking a SP with the Context. The component SecurityProfileManager is used for the activation and deactivation of Security Profiles. The SecurityProfileManager works as follow:

The MosesHypervisor is the component that acts as a policy decision point (PDP) in MOSES. The MosesHypervisor provides a central point for MOSES security checks against the policies defined for the active SP to regulate access to resources. The MosesHypervisor delegates the policy checks to its two managers: the MosesAppManager and the MosesRulesManager. The former is responsible for deciding which apps are allowed to be executed within a SP. The latter takes care of managing Special Rules [4].

The MosesPolicyManager 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. Moreover, this component also controls access to MOSES policy database (moses.db) allowing only applications with special permissions to interact with this component.

The MosesTaintManager component manages the “shadow database” which stores the taint values used by Taintroid [5]. In MOSES, we can taint specific rows of a content provider: to be able to perform per row filtering when an app access data in the content provider. For instance, it is possible to filter out from the query result data the rows which contain the information about device identifiers or user contacts. Given the fact that the enforcement of policies depends on the information provided by the MosesTaintManager, this component acts as a policy information point (PIP).

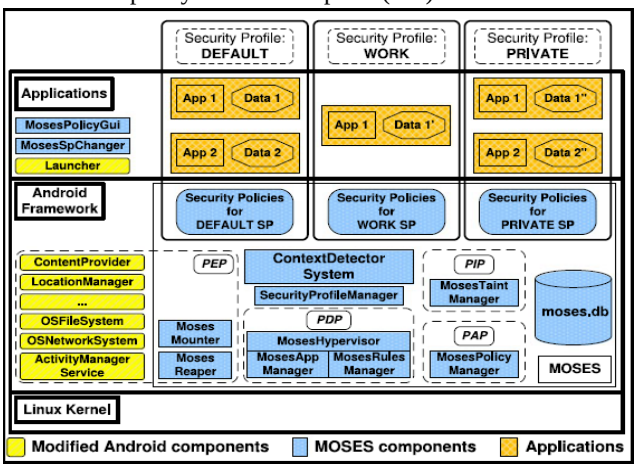


Fig. 2 Moses Architecture

The decisions taken by the MosesHypervisor need to be enforced by the policy enforcement point (PEP). MOSES affects several components within Android middleware where decisions need to be enforced. For this reason, the PEP includes several Android components offering system services such as LocationManager and ActivityManagerService. Moreover, some Android core classes (such as the OSFileSystem and OSNetworkSystem) are modified to enforce decisions regarding the access to the filesystem and network, respectively.

The enforcement of separated SPs requires special components to manage application processes and filesystem 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 we need to stop their processes. The MosesReaper is the component responsible for shutting down processes of applications no longer allowed in the new SP after the switch. In MOSES, applications have access to different data depending on the active profile. To separate data between profiles different file system view are supported. This functionality is provided by the MosesMounter. To allow the user of the device to interact with MOSES, we provide two MOSES applications: the MosesSpChanger and the MosesPolicyGui. The MosesSpChanger allows the user to manually

©IJRASET 2015: All Rights are Reserved

activate a SP. It communicates with the MosesHypervisor and sends it a signal to switch to the profile required by the user. The MosesPolicyGui allows the user to manage SPs [4].

B. Implementation

We have slightly changed the existing work of MOSES because the services required for it is very expensive. In base work author has created Context definition on which the activation or deactivation of security profiles depends. Each security profile is associated with one or more definition of Context. A context definition is a Boolean expression defined over any information that can be obtained from the smartphone's raw sensors (e.g., GPS sensor) and logical sensors. Logical sensors are functions which combine raw data from physical sensors to capture specific user behaviours (such as detecting whether the user is running). When a context definition evaluates to true, the SP associated with such a context is activated. It is a possible situation when several contexts, which are associated with different SPs, may be active at the same time. To resolve such conflicts, each SP is also assigned with a priority allowing MOSES to activate the SP with the highest priority. If SPs have the same priority, the SP, which has been activated first, will remain active. This behaviour required Google Maps service which is inexpensive. So what we have done is we are activating the profile based on the password which is given while creating security profile. If this evaluates to true, it will activate the particular profile according to it apps will be executed and its data can be accessed.

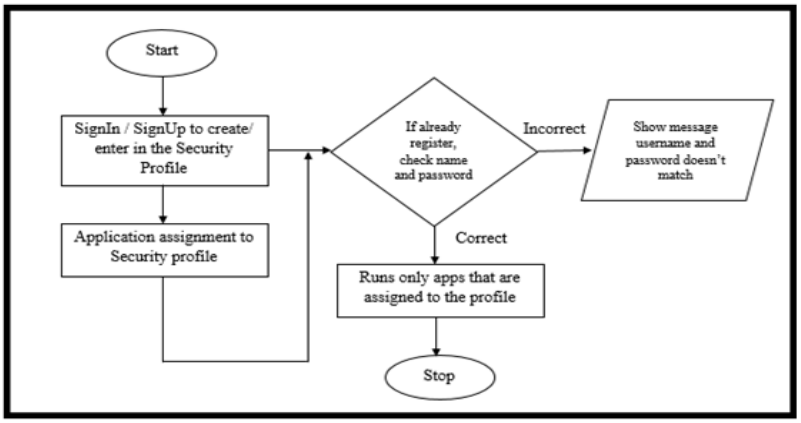


Fig. 3 Flowchart for MOSESApp

C. Performance Analysis of MOSESApp

Because of some limitations which are discussed in previous section, we are not implementing Moses in terms of context definition. So what we have done, we have control the activation and deactivation of profiles in terms of passwords. If login name i.e. context name and passwords matches then and then only particular profile gets activated else it will pop-up the toast like "Username and password does not match". Let's have a detail glance on the app. The very first step is to create context definitions which provides security environments or profiles to applications in order to do their executions in safe mode. Here context definition is defined in terms of username and password which can be also termed as "Login Credentials for Moses security environments". Security profiles or environments are the set of protocols to rules that regulates the app behaviours. Following screenshots gives detail description about Moses working:

1) As soon as application launches, it shows two button i.e. Sign-In and Sign-UP.



Fig. 4 Home-screen of MOSESApp

©IJRASET 2015: All Rights are Reserved

2) User has to create context definition by clicking on Sign-Up button. Assigns various applications to it in order to give them safe environment for execution. Here we are working on only seven application as you can see in the screenshot. We are assigning all apps to "HOME" profile for our convenience. User can assign anyone among these.
3) Again we have created another definition called as "WORK" to which none of these applications are assigned.

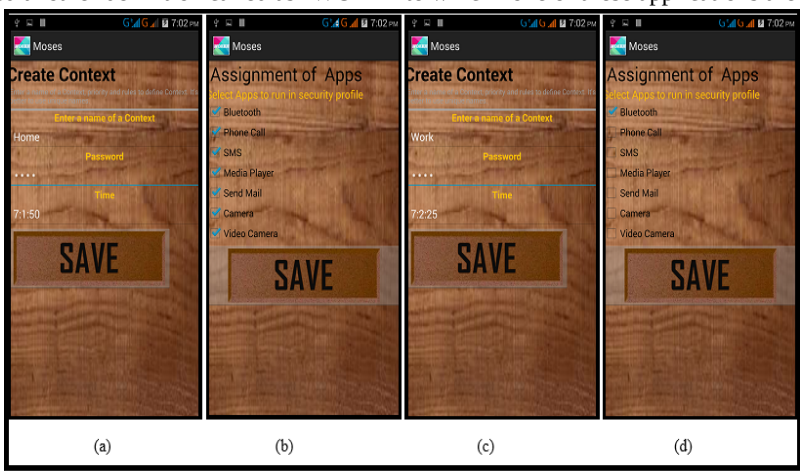


Fig. 5 (a) Context Creation (HOME), (b) Application Assignment (HOME), (c) Context Creation (Work), (d) Application Assignment (Work)

4) Again we have created another definition called as "WORK" to which none of these applications are assigned.
5) Once it is created, user can sign-in with anyone of this. If user sign-in with "HOME" then all the applications assigned with it will be executed on the other if sign-in with "WORK" then none of the applications will get executed.
6) With the help of Sign-In button user can manually switched between the profiles at any time. If user fails to enter correct username and password then it will show the message “Username and Passowrd doesn’t match”.

D. Performance Analysis of MOSESApp

To measure the energy overhead produced by MOSES and SecurePhoneApp, we performed the following tests. We charged the battery of our device to the 100 percent. Then, every 10 minutes we run seven third-party applications (sequentially) via a monkeyrunner script: Caller, Bluetooth, Email, Message, Media-player, Video-camera, and Camera. For each of them the script performed common operations representative for the applications (making calls, allowing and denying Bluetooth, sending e-mails and sms, playing music, capturing video and pictures).

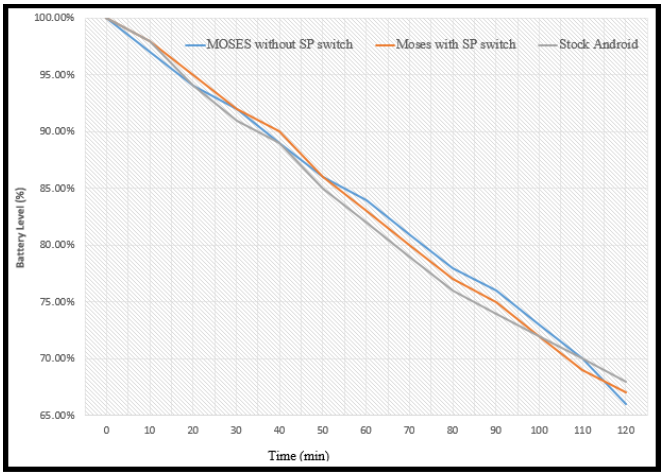


Fig. 6 Energy Overhead of MOSESApp

Each experiment lasted for a total of 120 minutes. We executed this experiment for three types of systems: Stock Android, MOSES without SP changes, and MOSES with SP changes (the system switched between two pro les every 20 minutes). During each experiment, every 10 seconds, our service measured the level of the battery and note this value. For each of the

©IJRASET 2015: All Rights are Reserved

three considered systems, we executed the test 10 times and averaged the obtained values. The results of this experiment are reported in fig. We note that the curves for the three considered systems behave similarly. This shows that the fact that MOSES is just running, or even switching between contexts does not incur a noticeable energy overhead.

IV. SECUREPHONEAPP APPLICATION

As discussed earlier, external security is provided by the SecurePhoneApp which works as follows. As soon as we install the application we have to activate all the service of DeviceManager. After that user has to set password for the device and also provides the number of attempts. Basically we do have 3 attempts so we will go for 3. As it is given in the fig. user has to enter correct password if fails, after 3rd attempt data will get wiped out. It may happen that unauthorized user is handling the device or device get stolen. So the first step of the theft is to take out the SIM-card so after taking out SIM all the data on the device will get format. But it also a possibility that an authorized user wants to change the SIM, so avoid formatting of data user can sign in as an authorized user. Also daily backup will be sent to authorize email-address.

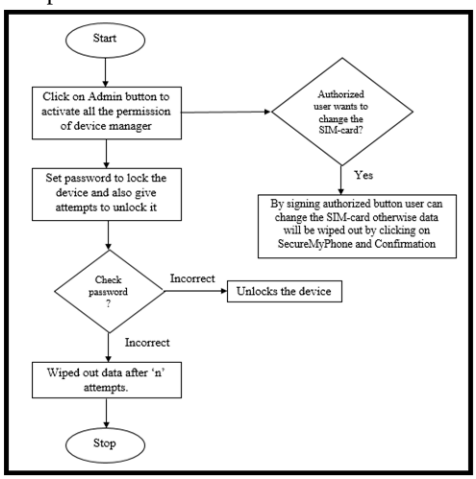


Fig. 7 Flowchart for SecurePhoneApp

So, the proposed work is all about how to secure the data on the device which is very sensitive. Many companies are willing to support employee-owned smartphones because of the increasing productivity of their employee. Hence, it is necessary to provide such kind of security to the data so that no one except authorize user can tamper with it accordingly.

A. *Working of SecurePhoneApp*

1) As we install the application it shows home screen.
2) User must click on "Admin" button in order to activate all the permissions which is given in the following figure. Clicking on activate button allow the application to work as DeviceManager for the device which controls all the permission about the device.



Fig. 8 Home-screen of SecurePhoneApp

3) User can set screen security by clicking on "Screen Security" button in order to lock home screen of the device. It allows to set any of the following:
   – None

©IJRASET 2015: All Rights are Reserved

– Slide
– Face Unlock
– Voice Unlock
– Pattern
– PIN
– Password

Also able to give 'n' number of attempts to unlock it. Normally, android mobile phone works on three attempts.

a) Another button is "Secure My Phone" which is used to give indication to the DeviceManger that the SIM-card has been changed and it's time to wipe out the device data. Also, if the user fails to enter correct screen security, after 'n' number of time all the data will be wiped out. By the time SecurePhoneApp creates ".zip" file of the folder in which user is storing their confidential data as well as send that ".zip" file to authorize user. This is done by clicking on "Confirmation" button which tells the DeviceManager to start the activity for clearing. Necessity to do so is that it may happen unluckily user has lost the device and someone else except the owner has got it.

b) Mostly people don't think to return it back. So what they do, they take out the SIM-card first and shut down the device about a month after start using it. And no one wants that someone else is tampering with their sensitive data. In short everyone in this world want to keep their data safe from Now-a-days, mobile device is not as important as the data present on the device. This problem is solved by "Secure My Phone" and "Confirmation" button. After starting the activity, the proposed app continuously check for the presence of SIM-card and does the action accordingly.

c) It may happen that owner of the device wants to change the SIM-card and to prevent from clearing out data "Authorized Sign In" button is there. By clicking on it, user can change the SIM-card without wiping the data.

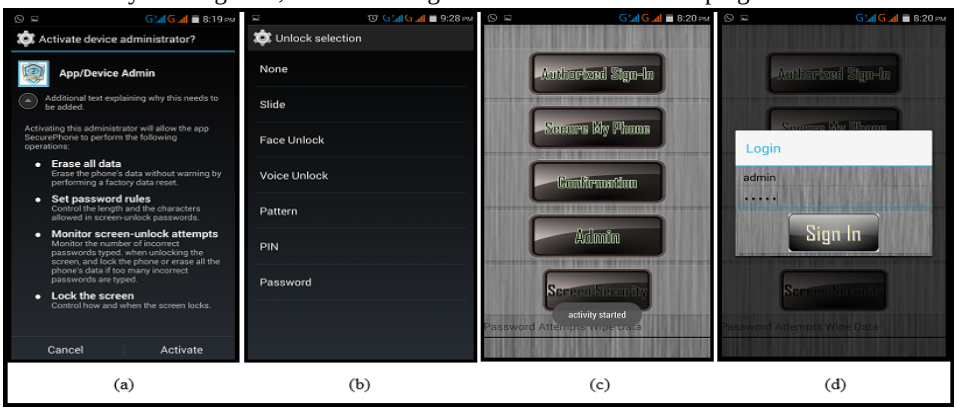


Fig. 9 Screenshots of SecurePhoneApp: (a) Device Administrator Activation, (b) Screen security to unlock device, (c) Confirmation to start activity of clearing data, (d) Authorised sign-in to remove SIM-card

B. User Rating based on Smartphone

The User Rating is given based upon the proper execution of both the application. The user rating shows that the applications prepared work well in different environment of android Operating system

TABLE I: USER RATING FOR SECUREPHONEAPP BASED ON THE SMARTPHONE

Sr. No.	Name of Mobile Company	Time required to generate ".zip" file	Size of ".zip" file	Time required to send Email
1	Micromax	10 sec	284.06KB	45 sec
2	Asus	12 sec	305KB	56 sec
3	Xiaomi MI3	25 sec	410KB	60 sec
4	Motorola	25 sec	1MB	65 sec
5	Lenovo	27 sec	1.5MB	70 sec
6	Sony Xperia	17 sec	500KB	57 sec
7	Samsung	10 sec	200KB	30 sec

©IJRASET 2015: All Rights are Reserved

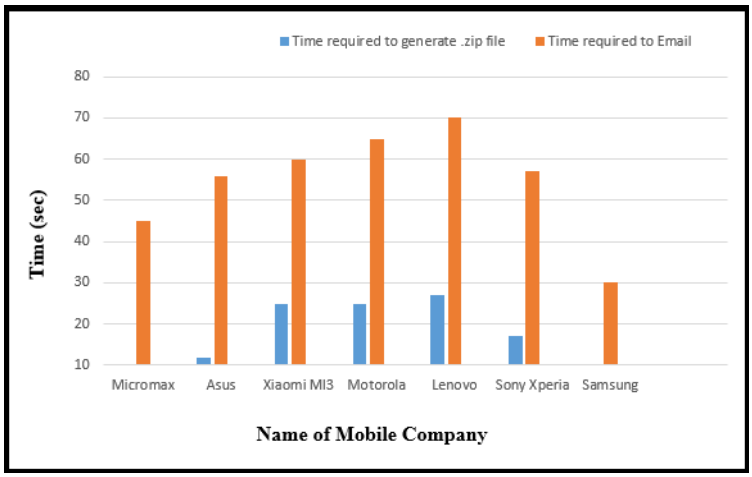


Fig. 10 User rating for SecurePhoneApp

From the above figure, it is clear that time required to create ".zip" depends on the size of the folder and time required to send email depends on the internet speed and the size of the ".zip" file.

V. CONCLUSIONS

To solve the problem of data being tampered by malicious app and by un-authorized user, the proposed application ensures that end-user can send daily backup to their authorized email-id. If they fails to enter correct login credential then data will be automatically emailed even in the second attempt of login credential. Also if someone stolen the mobile phone, theft will take out the SIM-card first then while taking out SIM data will get wiped out. The proposed work is related to how can we secure the data present on smartphone internally and externally not how to get mobile phone back. In today's world purchasing a smartphone is quite easier than losing the precious data.

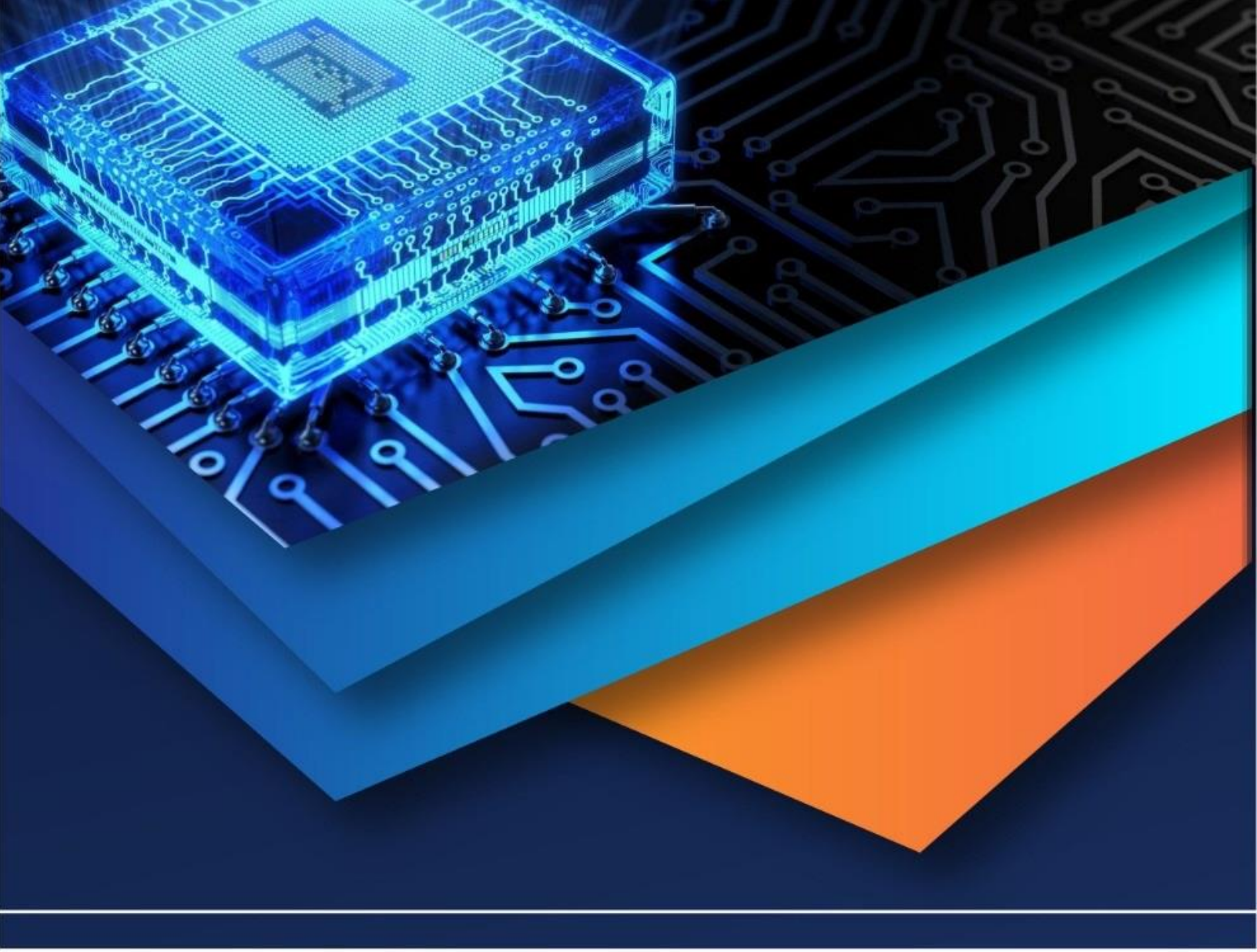
VI. ACKNOWLEDGMENT

We would like to show our sincere gratitude to them who directly or indirectly support us in completion of the manuscript.

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] http://www.s21.com/smartphones.html accessed on 17th Feb 2015.
[3] Y. Xu, F. Bruns, E. Gonzalez, S. Traboulsi, K. Mott, and A. Bilgic, “Performance Evaluation of Para-Virtualization on Modern Mobile Phone Platform,” Proc. Int’l Conf. Computer, Electrical, and Systems Science and Eng. (ICCESSE ’10), 2010.
[4] Yury Zhauniarovich and et. al., "MOSES: Supporting and Enforcing Security Profiles on Smartphones", IEEE Transaction on Dependale and Secure Computing, Volume-11, Issue-3, May-June-14, pp. 211-221.
[5] W. Enck, P. Gilbert, B.-G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, "Taintdroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", Proc. Ninth USENIX Conf. Operating Systems Design and Implementation (OSDI '10), 2010, pp. 1-6.

©IJRASET 2015: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)