Indian Journal of Science and Technology, *Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/102056, December 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Access Control Policy on Mobile Operating System Frameworks –A survey

G. Abirami[1] and Revathi Venkataraman[2]

Department of Computer Science and Engineering, SRM University, Kattankulathur – 603203, Kancheepuram District, Chennai, Tamil Nadu, India; abirami.g@ktr.srmuniv.ac.in, revathi.n@ktr.srmuniv.ac.in

Abstract

Background: Access control is the method of granting permissions according to policies. Mobile devices, such as Smartphone, are acting as multi-purpose devices for private as well as the corporate environment. Since the applications in the Smartphone are accessing in various contexts, there would be a leakage of data and applications. Furthermore, when third party apps are downloading on Android-based platforms, it causes threats to the existing system applications. So, different access control policies have been implemented in the Android-based Smartphone to separate the own application and corporate application for providing security. **Methods:** This paper presents a survey of access control models in various frameworks and compares them by their performance evaluations. The performance of each framework is corresponding to the characteristics of access control policies. These access controls are categorizing according to role, discretionary, mandatory, context, and attributes. **Finding:** We have found that, Context-Based Access Control (CBAC) and Dynamic Role Based Access Control (DRBAC) are providing better performance. Hence, to give robust security for mobile operating systems the hybrid access policies can be considered. **Applications:** This hybrid approach might provide a good Android security framework with acceptable performance.

Keywords: Access Control Policy, Android OS, Framework, Security, Smartphone

1. Introduction

Nowadays, the Smartphone plays a vital role in both personal and business purposes. Enterprises that allow employees to use the phones for the multi-purpose environment need to protect the information and applications on such devices and control their behavior. Furthermore, while the user installs third party applications on his mobile device where a malicious application may access email, SMS and MMS in which confidential data will be attacked, and if the device is lost or stolen then the enterprise data present in that mobile device will provide severe threats to company. Also, the data may be leaked accidentally due to combine of user's personal data and enterprise data. Personal devices that are using for working are connected to enterprise networks hence it is important to confirm that all mobile devices adhere with company security policies.

Although, to implement company rules on personal devices creates problem while incorporating a large number of devices. Also, it requires new security policies to manage with new security threats. Thus, effective security methods need constant updating of both devices.

Many access control policies are using the Android platform to enforce security policies. The access controls are Mandatory access control, Role-based access control, Context based access control, Discretionary access control, and Attribute based access control. In this paper, we will describe how these policies have been implemented in different Android platform frameworks to provide security for private mode and corporate mode isolations.

2. Various Access Control

Access Control is any security mechanism through which a system can able to access any data or do some action by granting or revoking the rights. To implement, we need to have an access control mechanism, subjects, objects, access procedure, and an environment condition (Figure1).

**Author for correspondence*

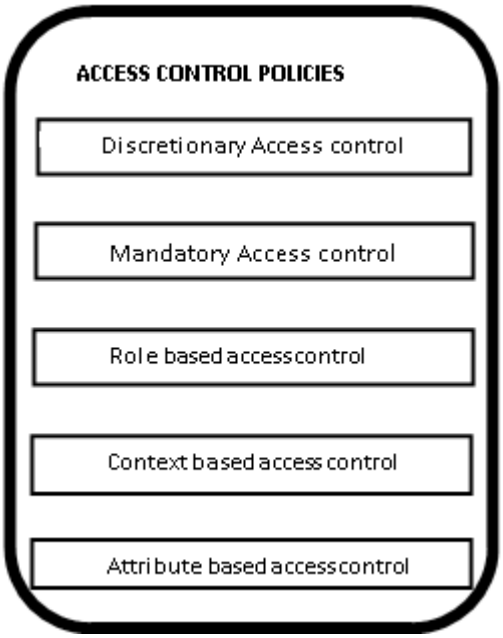

Figure 1. Access control policies.

- Subjects are the active entity such as user or process.
- Objects are entities such as files, directories, network connections, shared memory segments, IO devices, etc.
- Access Procedures are grant/revoke/deny permissions.
- Environmental conditions are influential factors that are autonomous to subject and object that may be used as attributes at decision time to affect an access decision. The Environment attributes might be location, time security threat level, temperature.
- An operation is a function of a subject upon an object. The various Operations are read, write, edit, delete, copy, execute, and modify.
- The policy is a set of rules or relationships that define the set of proper operations and the subject may perform some operation on an object in suitable environment conditions.

The various access controls are following:

2.1 Mandatory Access Controls (MAC)

With Mandatory Access Control, the security policies are providing in system administration level rather than user level. So, the users are not able to change the security policies. The MAC operating system limits the ability of a subject or user to access control and perform some task; objects contain a set of security attributes. Whenever a user wants to access an object, an authorization rule applied by the operating system kernel explores these security attributes and decides whether the access can take place. Any operations of the subject on any object are tested against the set of authorization rules to determine if the functions are allowed (Figure 2).

2.2 Role Based Access Control (RBAC)

Access decisions in RBAC based on the roles that individual user have as a part of an organization (Figure 3). The user is assigning to different positions, with permissions to use the resources. Hence the person has the least privilege based on their role to access the sensitive data.

2.3 Discretionary Access Control (DAC)

Discretionary Access Control (DAC) allows the users and Administrators to define an Access Control List (ACL) on specific resources/objects (e.g., Files, Network socket, Database, etc.). The resources can be accessed by the subject only when the owner provides permission called discretion

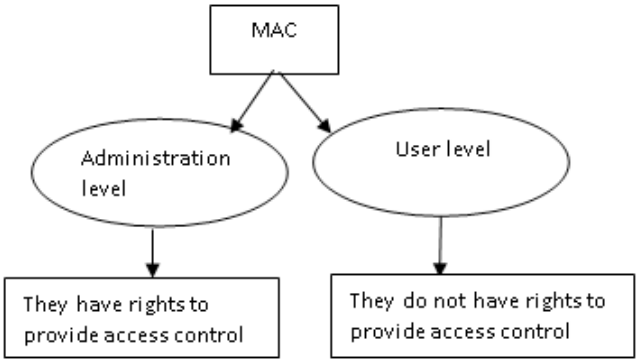

Figure 2. MAC.

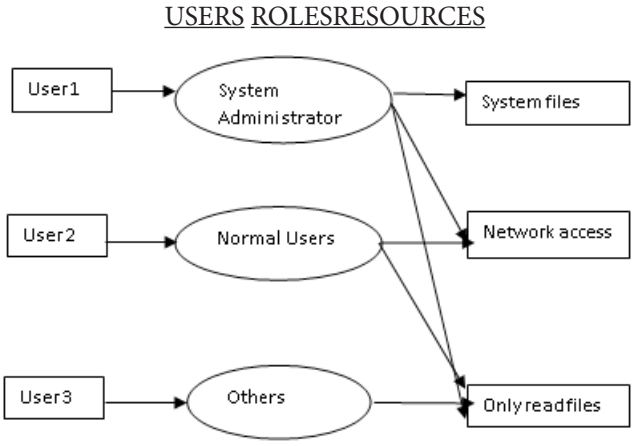

Figure 3. RBAC.

(Figure 4). Here, Windows, Linux, Macintosh, and UNIX are work on DAC models. If the user creates any files, then they have the rights to provide access privileges to other users on that file. Hence if the user wants to access this file, then the access control decision is done by operating systems. Discretionary Access Control are providing with the high degree of flexibility to the large variety of application.

2.4 Context Based Access Control (CBAC)

A Context-based access control mechanism gives privileges with dynamically granted or revoked to applications which on the particular context of the user (Figure 5). Context has physical context are location, time and online are associate a physical sensor device such as Clock, GPS, Bluetooth, etc. The logical context defines functions over physical sensors (i.e. the user answering the phone calls).

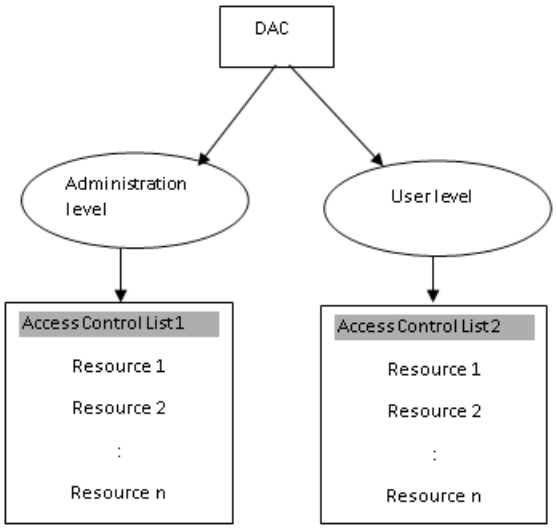

Figure 4. DAC.

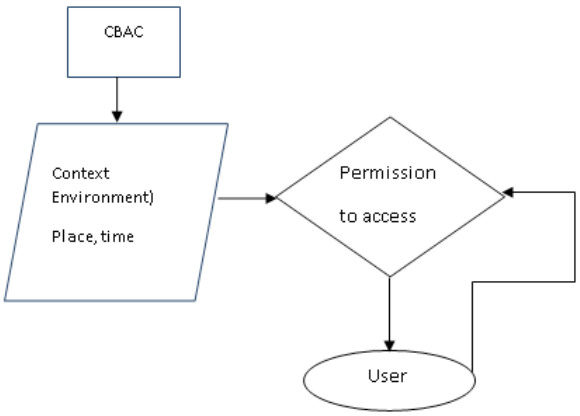

Figure 5. CBAC.

2.5 Attribute Based Access Control (ABAC)

In Attribute based access control the rights are granting to the users based on attributes which combine with the policies. Authorization relies on a set of operations is determined by evaluating the attributes associated with the subjects, objects, and requested services. Hence ABAC assigns a policy with abject and Environment (Figure 6).

3. Mobile Frameworks with Different Access Control

3.1 FlaskDroids

Flaskdroid[1] is a generic framework implemented on Android-based mobile devices that operate on middleware and kernel layers which provide Mandatory Access Control (MAC). The goal of MAC is to give flexibility in security policy based on user decisions. Secondary goals of the design include application transparency, security- in- depth, ease of guarantee, and minimal performance impact.

In FlaskDroid the security policies depend on objects and subjects of the context in runtime. Also the security policy of app developers, end-user, or third parties is managed by the different security server. Hence, FlaskDroid's middleware and kernel level are designed fully with fine-grained and high flexible access controls for an operation.

In FlaskDroid, the policies are assigning by User Space Object Manager (USOM). These policies are system policy, application developer policy, and user policy. The Access controls policy are all - allow, any - allow, and priority. The all-allow policy allow access of all stakeholder's policies. Whereas, in any - allow only one stakeholder policy with

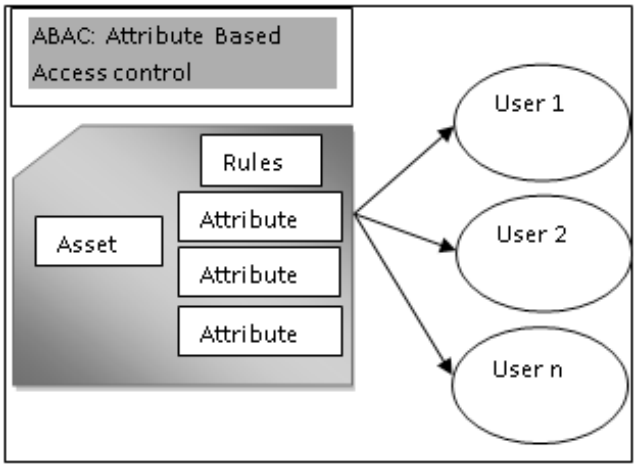

Figure 6. ABAC.

higher priority is allowed to access. The FlaskDroid are purely working on type enforcement policy. The type enforcement is subject type (e.g., Calling application), the object type (e.g., email of contacts), the class type (e.g., data of contacts), and the operations (e.g., Query).

The FlaskDroid architecture is evaluated by; the user was asked to use the pre-installed system apps such as maintaining contacts, writing SMS, browsing the Internet, and sharing data. For testing purpose the Galaxy Nexus device with the FlaskDroid operating system are provided to users. It contains the only subject and object. Based on testing it was found that 109 access control rules were providing and they checked the functionality of FlaskDroid with free apps using a synthetic test app that provides information and unmodified apps from Google play including WhatsApp Messenger and Facebook apps. In all these applications they have found that the policy of FlaskDroid successfully prevented from accessing the private information.

The performance overhead of the Flaskdroid operating system with no-allow policy in Samsung Galaxy Nexus mobile device is evaluated by the author. They have compared their permission check with vanilla Android 4.0.4 version. The Flaskdroid execution time is 329.505 μs which are less than vanilla Android execution time (330.800 μs). The memory space occupied by Flaskdroid with no-allow-rule is 15.673, and basic policy is 16.184, and vanilla Android permission check is taking 15.985. Hence, in Flaskdroid both time and memory overheads are accepted. Furthermore, Flaskdroid designed with two layers, MAC framework in Android platform, which provided different security type for business and private application at installation time.

3.2 DRBAC Architecture

To enhance the Android security in business environment Dynamic Role Based Access Control (DRBAC) model of Android[2, 3] is highly used. This system provides security in both application level and permission level on android mobile device rather than workstations with multi-user capability according to fine grained role based access.

Also, it leverages both roles based and context-aware capabilities of mobile devices. Even it has flexible access control and it is able to reduce most dangerous security threats to mobile devices. The DRBAC can be used in a large business enterprise easily with versatile mobile devices. Industries uses role based access control due to its least privilege principles and limiting the access of resources based on their roles. The Android version 4.0.4 (Ice-cream sandwich) are modifying with DRBAC with minimum number of existing android codes. Hence DRBAC provides unlimited access to any permissions request. Although the modifications of the Android operating system needs to change.

Experiment in Galaxy Nexus for Doctor and patience's role. Depending on the functions the applications on the mobile device can be accessed (e.g., The Doctor and nurse could be able to access medical information. But, the patient does not have rights to access it). The energy consumption was less due to denial of Internet, send_ SMS, email during working hours. Also, if the users have any roles or in context their information are loaded into memory even more users are involved with limited policy. The effectiveness of DRBAC is according to security policy used. Although, a simple security system are strictly restrict the threat of malicious and vulnerable app.

The time overheads of DRBAC range from 400 to 800μs with Android own permission checking expenses of 630μs. Hence the overall cost has been reduced by a considerable amount.

3.3 Context Droid

CREPE[4-7] is a fine-grained context related policy enforcement system for Android. Through some app, SMS and QR code the authorized third party and user can set this policy. This architecture is context related policy enforcement for android operating systems. It separates the system into many loosely coupled modules which provide high flexibility for usage and openness for other developers.

Hence, any developer can design a new software sensor with high level of security to implement specific needs such as the user is in crowded places. CREPE supports both low-level and high-level sensors. The low-level sensors are physical context parameters, such as location, time, temperature, etc., which are associated with physical sensor devices (i.e., clock, GPS, Bluetooth, etc.). The high-level sensors are related to logical sensor and the context that are defined by function over physical sensors. The user running in an open environment or the user is answering any phone calls are examples of logical sensor.

CREPE provides user for defining their context related policies on the mobile phone during at runtime. Context related policies composed of access rules and actions they are in context database. Also, based on the priority the policy and regulations (ACESS/DENY) are provided.

The User level, the Framework level, and the Kernel level are three levels of abstraction on android software stack where CREPE operates. Hence, their framework changes its policies at the application during execution time and not only at installation time. Also, the administrator can able to add, delete and modify new context or policies at run time. Therefore, the system may change its existing connection to new services, i.e. the internet connection may be disabled and new services could be connected while the user enters the meeting room. Thus CREPE regulates the behavior of mobile phone based on the contextual situations.

The author used Google Dev3 Phone for experiments, to evaluate performance they ran some tests on the Stock Android phone and Dev3 phone with CREPE. They have tested for every 10 minutes, such as making the phone to start a call with every 10 minutes and lasting a call for 110 seconds, then starting and ending some set of application such as MMS, Calendar, Email, and Gallery in Stock Android. Then, they have done the same experiment on Dev3 phone with CREPE and found time overheads. The CREPE contains different rules (i.e., 15, 30, and 60). The CREPE with 60 rules has taken more process time. It that with the increase in standards and policies (e.g., 60 policies) time and storage requirement also increased.

Energy overheads have calculated in both Stock Android and CREPE system. They have automatically placed a phone call and stating some application to run on every 10 minutes and allowing them over a period of 120 minutes. They have repeated these experiments for ten times in both stock android and CREPE system with 15 current rules. The battery started at 4.50mV, in Stock Android, it reduced to a 4.058 and CREPE system with the same 4.50mV of the battery was reduced to 3.950mV. Since it depends on context, such as GPS the energy consumption was more in CREPE.

3.4 MOSES Droid

MOSES[8, 9] is a policy-based framework for separating modes (private/personal data) by implementing with software virtualization techniques for isolation of data. The workspace is divided into two portions. The first part could be utilized for sensitive/corporate and trusted applications. The second part contains entertainment data where third party games and traditional application can install. Software isolation by enforcing Security Profiles (SP) which has been implemented based on contexts and attributes. The important characteristics of MOSES provide dynamic switching of applications between different security profiles. Each security profiles are done in fine-grained to a level of an object (e.g., SMS, file) and the single application. Within each security profile, MOSES applies an attribute-based access control model and the users can set their fine-grained access control policies to control application behaviors. The attribute types are Subject, Operation, Taint, Target and SP-Name where Subject is an application in which rules can be applied. The operation is the action that the subject is executing; Taint provides detail to hide sensitive data; Target provides the resource is accessed. The sp-name attribute is meant the security profile name in which the policy is viable.

They have a set of the policy rules as ALLOW, DENY, and ALLOW_WITH_PERFORM. The experiments have done with Google Nexus Phone. To evaluate the performance battery is charged to 100 percent. Then, for every 10 minutes, they ran four applications such as Browser, Contact, and Email, calendar sequentially. They tested their applications with Stock Android, MOSES without SP and MOSES with SP. The system energy overheads ware same in all these framework models. Because of duplication of data for different profiles, storage expenses incurred. The time requirement is based on various numbers of users and special rules on application. The switching time is increasing when the number of user applications increases.

Thus, MOSES considered to a first software solution of policy-based security model. But it has some limitation due to the same UID using for some applications. Hence, the MOSES rules and restriction apply in one application can also be extended to other application with the same UID. So, the security profile activated for one application would also be allowed another application which causes energy overheads.

3.5 DAC framework

Discretionary access control[10] is a traditional access control that implemented with MAC in the security framework of android operating system. The framework is developed and deployed by the various security experts in the enterprise. Also, they ensure that to access an application the user should have the necessary rights (i.e. Permission) in DAC[11]. Providing special security rights removed[12-14]. They have experimented this with Nexus 7 and altered the Android code. Hence, it is negligible overheads.

The merits of DAC are providing least privilege access of an object by an individual user with limited rights. It is

Table 1. Performance of various access control policies.

Mobile Frame-work	Access control policy	Energy consumption	Time	Storage Over-heads	Access control Advantage and Disadvantage	
Flask–Droid	Mandatory access control	Moderate	Acceptable	Acceptable	1. Strict control over information flow. 2. Strong exploit containment	1. Major usability problems. 2. Cumbersome administration.
DRBAC	Role based access control	High with acceptable	Low	Low	1. Great efficiency.	1. Cannot specify fine grained rules.
Context–droid	Context based access control	High with acceptable	Acceptable	Acceptable	1. Transparency, Flexibility.	1. Still static model.
Moses–droid	Attribute based access control and Context based	High with acceptable	More (switching to profiles)	More (Data duplications)	1. Dynamic 2. Fine grained Rules	1. More Effort to define policies.
DAC	Discretionary access control and MAC	Low	Less	Low	1. Simple and efficient access rights management 2. Scalability	1. Intentional abuse of access Rights. 2. Unintentional abuse of access rights. 3. No control over information flow.

very cost effective for home and small business since its implementation is invisible to the user. But, security verification and maintenance of the system is very difficult in DAC due to its access rights of user with own object.

Table 1 shows the performance of various access controls. From this, FlaskDroid performance overheads are low with better information flow control, but, difficult with administration level. Dynamic Role Based Access Control (DRBAC) has provided less time and lower storage overheads with high efficiency, but it did not specify fine-grained rules. Context droid frameworks gave acceptable expenses with transparency and flexible controls. Moses droid is the dynamic model. However, it contains more defined policies.

Due to less information flow control of DAC and make efficient usage it should combine with other policy frameworks. The Android framework with different access control policies discussed in Section III. The different access control policies have been used to ensure the security of Android-based frameworks. Each access control policy has its characteristics. Hence, if we leverage on these access control policies, we can get their functionality also advantages to provide a hybrid framework for Android with improved performance. Since Dynamic Role Based Access control (DRBAC) has less time and storage overhead, it can combine with Context-Based Access Control (CBAC) which has context information and Attribute-Based Access Control (ABAC) which has more security policies. This hybrid approach might provide a good Android security framework with acceptable performance overheads.

4. Conclusion

With rapid increases in mobile devices such as Smartphone's and ability to use as both personal and corporate applications, it is necessary to provide security to the device. In this paper, a survey on various access control policies has presented. Moreover, we have discussed how these policies are implementing in different Android based frameworks for providing security in personal data and corporate data. The Access control policies are Discretionary access control, Mandatory Access control, Context-based access control, Attribute based access control, and Role based access control. Also, based on this access control policies the performance evaluation of the Android frameworks has provided. We have found that Attribute and Context based policies are providing more security for modes of separation in Smartphone's. Hence Context-Based Access Control (CBAC) and Attribute-Based Access Control (ABAC) can combine with Dynamic Role Based Access Control (DRBAC) which has less time and storage overhead to provide a hybrid framework. This hybrid approach might provide a good Android security framework with acceptable performance costs. So, we

would concentrate on this hybrid policy for providing robust security in the mobile OS framework.

5. Acknowledgement

We would like to thank the SRM University for providing infrastructural facilities for doing research work.

6. References

1. Bugiel S, Heuser S, Sadeghi AR. Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies, Proceeding 22nd USENIX Conf. Security 2013.
2. Jun Zheng, Qikun Zhang, Shangwen Zheng, Yuan Tan. Dynamic Role Based Access Control Model Journal of Software, 2011 Jun; 6(6). Doi: 10.1145/2523649.2523676.
3. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-Based Access Control Models, [C] IEEE Computer. 1996; 29:38–47. Doi: 10.1109/TDSC.2010.55
4. Bilal Shebaro, Oyindamola Oluwatimi, Elisa Bertino. Context-Based Access Control System for Mobile Devices, IEEE Transactions on Dependable and Secure Computing. 2015 March/April; 12(2). Doi: 10.1109/TDSC.2014.2320731.
5. Conti M, Nguyen VTN, Crispo B. Context-Related Policy Enforcement for Android, Proceedings 13th Int. Conf. Inf Security. 2011; 331–45. Doi: 101109/TIFS2012 2204249.
6. van Wissen B, Palmer N, Kemp R, Kielmann T, Ba H. Context Droid: An Expression Based Context Framework for Android, Procerding Phone Sense'10. 2010; 1-5.
7. Bai G, Gu L, Feng T, Guo Y, Chen X. Context-Aware Usage Control for Android, Proceeding Int'l Conf. Security and Privacy in Comm. Networks (Secure Comm '10). 2010; 326–43.
8. Russello G, Conti M, Crispo B, Fernandes E. MOSES: Supporting Operation Modes on Smartphones, Proceeding 17th ACM Symp, Access Control Models and Technologies (SAC MA T'12). 2012; 3–12.
9. Russell G, Conti M, Crespo B, Fernandes E. MOSES: Supporting and Enforcing Security Profiles on Smartphones, IEEE Transactions on Dependable and Secure Computing. 2014 May-Jun; 11(3). Doi: 10.1109/TDSC.2014.2300482.
10. Michael Backes, Sven Bugiel, Sebastian Gerling, Philipp von Styp-Rekowsky, Android Security Framework: Enabling Generic and Extensible Access Control on Android. arXiv: 1404.1395v1 [cs.CR] 4 Apr 2014.
11. Bugiel S, Davi L, Dmitrienko A, Heuser S. Sadeghi A-R., Shastry, B. Practical and lightweight domain isolation on Android, Proceeding 1st ACM Workshop on Security and Privacy in Mobile Devices (SPSM '11) (2011), ACM, Doi: 10.1145/2046614.2046624.
12. Beulah Hemalatha S, Vigneshwaran T, Jasmin M. Security Comparison of Android and IOS and Implementation of user Approved Security (UAS) for Android. IJST. 2016; 9(14). 9i14/87071, Doi: 10.17485/ijst/2016/v9i14/87071.
13. Shaukat Ali, Shah Khusro. Mobile Phone Sensing: A New Application Paradigm. IJST. 2016 May; 9(19). Doi: 10.17485/ijst/2016/v9i19/53088.
14. Pandey M, Rajashekar Babu M, Manasa J, Avinash K. Mobile Based Home Automation and Security System, Indian Journal of Science and Technology. 2015 Jan; 8(S2):1216. Doi: 10.17485/ijst/2015/v8iS2/57792.