ISSN 2348-375X

Unique Journal of Engineering and Advanced Sciences

Available online: www.ujconline.net

Research Article

DYNAMIC ACCESS CONTROL SYSTEMS FOR MOBILE APPLICATIONS

Vinoth Kumar J[1*], Vikramarajan Jambulingam[2]

[1]Department of Computer Science and Engineering, Bhajarang Engineering College, Tamil nadu, India
[2]Department of Electrical and Computer engineering, University of Gondar

Received: 28-04-2015; Revised: 26-05-2015; Accepted: 24-06-2015

*Corresponding Author: **J. Vinoth Kumar**
Department of Computer Science and Engineering, Bhajarang Engineering College, Tamil nadu, India, Mobile: 09677823604

ABSTRACT

Mobile applications mostly have access to susceptible data and resources on the user equipment. Abuse of this data by malicious applications may lead to privacy leaks and sensitive data outflow, for example, malicious application clandestinely recording about a secret business discussion. This occurs due to the fact that users of Android do not have control over the capabilities of the applications once they have been issued the requested rights during installation. In most of the cases, if an application gets a rights based on the specific user context and thus we can have a context-based access control mechanism by which rights can be dynamically decided or canceled to applications based on the specific context of the user. This paper proposes basically such an access control mechanism. Our implementation of context distinguishes between directly located sub-areas within the same location. We have changed the Android operating system so that context-based access control limits can be precise and imposed. We have conducted some experiments to review the effectiveness of our access control mechanism and the correctness of context discovery.

Keywords: Context-based access control, Security and confidentiality, Context-related policies, Mobile apps.

INTRODUCTION

As mobile phones are turning increasing powerful in terms of computational and communication capabilities, application creators are taking advantage of these capabilities in order to give new services to their applications. For example, on the year of 2013 Samsung opened its Galaxy S 4 device with 8 CPU cores and 9 sensors that enhances the device with powerful resources. The many of these resources can gather sensitive data and may reveal users to high security and privacy threats if applications use them inadvertently and without the user's knowledge. The risk rises when a device application runs maliciously and makes use of device resources to discover on the user or disclose the user's personal data without the user's wish[1-2]. To stop such risks[3], users should be able to have an increased control over their device capabilities by dropping certain application rights while being in susceptible contexts e.g. confidential meetings. Since such a function is still not present in popular Mobile phone systems, such as in Android systems, it is important to investigate ways for providing such an ability to control devices. Context-based policies are also a requirement for politicians and law enforcement people who would require disabling camera, microphone, and location services from their devices during confidential meetings while getting back these resources in non-confidential locations. Earlier works on security for mobile operating systems concentrates on limiting applications from accessing sensitive data and resources, but mostly lacks competent techniques for imposing those limits according to fine-grained contexts that distinguishes between closely located subareas[4]. Existing location-based policy systems are not good enough to make a distinction between nearby locations without extra hardware or location devices[5-7]. The design of context based policy systems for Mobile phones is demanding as it should fulfill the following requirements:

1) Applications should not have the ability to fake the location or time of the device, as they must not be able to crossover the policy limitations functional on the device in a specific context.
2) As users are thought to be mobile, the policy limitations should get applied automatically on the device as the device's location changes.
3) The accuracy of location needs to be better than the location accuracy by GPS, as we require applying different policies in different locations or nearby sub-areas located within the same GPS location.

4) The enforcement of context-based policies must not make the application developers to modify source code, or impose any additional prerequisite on their applications.

We propose a context-based access control mechanism for Android based mobile systems that permit Smartphone users to create configuration policies over their applications' usage of device resources and services at different situations. Through the CBAC mechanism, users can set restricted rights for device applications when using the device at workplace and device applications can re-gain their original rights when the device is used at house. This change in device rights is by design applied as soon as the user device suits a pre-defined context of a user-defined policy.

We describe context according to location and time. Location is found basically through visible Wi-Fi access points and their particular signal strength values that permit us to distinguish between nearby sub-areas within the same work space, in addition to GPS and cellular triangulation coordinates if available. Users can even be more accurate by distinguishing between sub-areas within the same location, like living rooms and bedrooms at house or meeting rooms and offices at workplace. Once the user sets the device policies that describe device and application rights according to context, the policies will be repeatedly applied if the user is within a pre-defined physical location and time interval.

This paper is structured as follows. Part 2 gives basic concepts and background details. Part 3 describes the design of the architecture and explains our access control mechanism. Part 4 explains our policy constructs and their classification followed by implementation and other technical aspects in Part 5. Part 6 explains our technique in managing context information and how we keep policy limitations appropriate with device location. Part 7 reports results of some experiments to ascertain the accuracy of context information and the impact of policy limitations on android applications. Parts 8 discuss related work and gives future work, correspondingly.

BACKGROUND

In this part, we discuss related background information on Android operating system and its various access control techniques, and some basics on location services.

Android

Operating System

The Android operating systems are developed from Linux based kernels and have increased support for security[8] and confidentiality[9].

Authorization System

The Android authorization system controls which application has the rights of accessing certain device resources and data. But it can raise the dangers of revealing the users' data and raise the impact of a bug or susceptibility. Each application maintains the Authorizations listed in its AndroidManifest.xml file at the time of installation, and users have to either award all the requested Authorizations to continue with the installation, or stopping the installation.

Android Application Components

Every Android application consists of four required components: Activities, Services, Content Providers, and Broadcast Receivers. An Activity describes an application's user interface. The Service component is created to be used for background processing. The Content Provider component performs as a global instance for a specific application, so that all applications of the phone can use it. It stores and manages a shared set of data belonging to the owner application using a relational database interface. Finally the Broadcast Receiver component acts as mailboxes that make applications to request for system or application events.

Intents

Intents are asynchronous messages that permit the three core application components such as Activities, Services, and Broadcast Receivers to communicate between each other.

Location Gathering Methods

In this work, we depend on Wi-Fi-based locationing techniques to retrieve the location of the device. Furthermore, we use and collect location data retrieved from GPS and cellular networks for cases where there is no Wi-Fi coverage in the areas concerned.

GPS based location Service

The Global Location System (GPS) is a locationing tool present in most mobile phones which uses data signals from satellites to calculate the location of the device. Data received from satellites has the time stamp of sending, the orbital information and the location of satellites. The location information provided from the GPS includes the latitude, longitude, altitude, and time. The accuracy of this method is expected to be in the range of 50 to 75 meters.

Cellular based Location service

Cellular triangulation is another locationing approach based on cellular technology. It is means of tracking a mobile phone's current location using radio towers. Through contacting every nearby antenna tower, cellular triangulation can make a dimension of how far away the cellular mobile device is based on the signal it is transmitting. This is completed by measuring signal strength and the round-trip signal time. The accuracy of this technique changes according to the density of cell towers existing in particular area.

Wi-Fi based Location Service

Many mobile phone devices nowadays depend on Wi-Fi-based locationing techniques due to its efficiency in calculating the location of a device especially in places when GPS and cellular signals are not available and weak[10]. These mechanisms are based on comparing the device's received Wi-Fi access points with database fingerprints containing Wi-Fi access points with known geographic locations.

SYSTEM ARCHITECTURE

Our structure consists of an access control mechanism that details with access, collection, storage, processing, and usage of context information and device policies. The Context Provider gathers the physical location values through the sensors of the devices and stores them in its own database, linking each physical location to a user-defined logical location. It also verifies and updates those values whenever the device is re-located.

The Access Controller reins the authorizations of applications and stops unauthorized usage of device resources or some services. Even though the Android OS has its own Authorization control system that verifies if an application has rights to request resources or services, the AC complements this system with more controls and specific customized authorizations that better show the application capabilities and narrow down its accessibility to resources. For example, the Authorization READ_ PHONE STATE gives rights applications a set of information such as the phone number, the IMEI identifier, subscriber identification, phone state, SIM serial number, etc.

The Policy Manager represents the interface used to set policies, particularly assigning application limitations to contexts. It predominantly gives control to the user to configure which resources and services are accessible by applications at the given context given by the context provider. For example, the user through the policy manager can set a policy to enable location services only when the user is performing work during weekdays between 9 am and 4 pm.

The Policy Executor imposes device limitations by comparing the device's context with the set policies. Once an application requests access to a resource or service, the policy executor checks the user-configured limitations set at the PM to either issue or to ignore access to the application request. The PE acts as policy enforcement by sending the authorization information to the AC to handle application requests, and it has the responsibility to resolve policy conflicts and apply the strictest limitations.

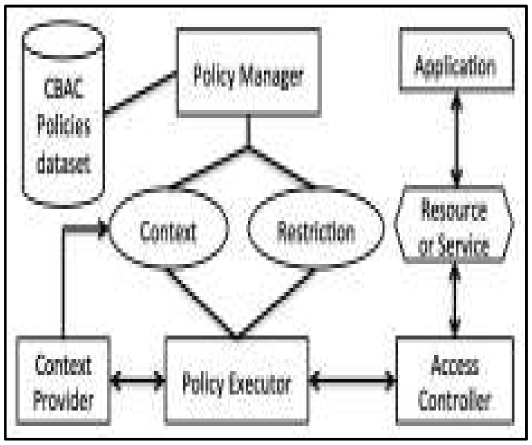

Figure 1: Access Control

Mechanism

Restriction: Let s∈SUB, o∈OBJ and a∈ACTION. A policy restriction is defined as the tuple [s,o,a] such that

a=revoke_Permission if type(o)=Permission

a=shadow_Data if type(o)=Data

a=disable_Intent if type(o)=Intent

a=Save_State if type(o)=System_Peripheral

Context: Let LOC be the location data of a particular sub-area and Name be a user-defined logical location that corresponds to that particular sub-area. And let {ST, ET, R} correspondingly be the starting time, ending time, and frequency to when a particular policy should be imposed. A policy context is described as the tuple [LOC, {ST, ET, R}] such that

LOC = <Name>, <latitude, longitude

<Wi − Fi − APs, RSSImin, RSSImax>

Policy: Let r be a limitations as defined above and c be a context as defined above. A policy is defined as a tuple [r, c].

Below is an example of a policy that disables the Skype application from accessing the Camera Authorization monthly between 6.00 pm and 8.00 pm on the second of every month starting August 1, 2014 at our department meeting room.

POLICY CATEGORIES AND EXAMPLES

We classify location based run-time policies according to the type of limitations and changes that we need to apply on the Android OS. Resource Limitations Policies: As Android apps are issued all their set of requested authorizations upon installation, this group deals with policies that can manage these authorizations and control the applications' access to system resources after installation. Authorization limitations are either applied system-wide or per application, as it may be possible to cancel some or all of the formerly issued authorizations per application. These authorizations are managed by intercepting an application's request to use a authorization at run-time, and then detect if the Authorization should be issued or denied depending on the current context.

Data Access Policies: This group relates to user data stored on the device such as Contacts, Calendar, Accounts, etc. As device users may want to disable their applications from accessing such important data at specific contexts, we design data-specific CBAC policies that permit users to set limitations on data access per application or system-wide.

System Peripheral State Policies: This refers to policies that limit applications from changing the state of the device peripherals such as switching the Bluetooth state into either on or off. Limitations on the system peripherals are implemented mostly through modifying the peripheral API method to intercept any applications request to enable particular system peripherals.

Multitasking and Intercommunication Policies: This refers to policies that limit data communication and messages between different devices applications in which many systems limitations can be applied.

User Security Policies: This group of policies relate to changes of the security level of the device which permits users to specify security limitations based on context. With these policies, users can mention the context in which to ask for a password for using the device. Users can also select to block certain applications from loading and running at a specific situation, while allowing them at different contexts.

SYSTEM IMPLEMENTATION

In this part, we introduce the technical details of our implementation which includes our changes to the Android OS and the components of the Policy Manager custom application that acts as an intermediary between the OS and the user's desired policy configurations.

Policy Manager Components: The Policy Manager custom application consists of the four Android application components: Activities, Broadcast Receivers, a Content Provider, and a Service.

Activities: The user interacts with the Policy Manager through activities, and through these activities, a user can be able to define physical locations and subsequently configure a set of policies for these locations. The elements of these activities include Application Events, Authorization Access, Resource Access, System Preferences, and Time Limitations.

Broadcast Receiver: We extended the Android's Broadcast Receiver class and created two more new custom classes such as the start Location Service Receiver and the Boot Receiver classes.
Service: Location Service finds if the device has moved to or still is in an already registered area. Offloading the accumulation of location-based data in a separate thread limits the performance impact of the execution of the Location Service on the Policy Manager. We use the Alarm Manager to occasionally trigger the Location Service to make sure the device's location is always up-to-date. By defaulting, the Location Service is started once per minute, but we give the user the option to configure how often the service is run.
Content Provider: The policies set by the user are stored within the Policy Manager data directory.
Authorization Management: In the Android system, all resources that need explicit access rights in the form of authorizations are confined by the Activity Manager Service class via authorization verification. When an application tries to use any of these resources, the Activity Manager Service's method called check Component Authorization is called to verify if the calling application has the appropriate authorizations to access the resource.
Limitations on User Data: We complicate user data from applications trying to access it if the policy limitations apply to those applications. We change the Android APIs that access the user data saved on the device.

CONTEXT MANAGEMENT

Location Capturing Stage: Location information gathered from GPS and cellular towers is also collected to our context description but may not be enough for indoor localization particularly that they may become not available inside buildings or areas within building structures. A spatial region is indicated by combining together GPS coordinates, cellular location data, and Wi-Fi APs and signal strengths. In Android, the GPS coordinates and cellular location data are gathered in a similar fashion by calling the Android Location Manager Service. Once the Location Manager is called, we request location updates by calling the request Location Updates method that precedes a Location object which consists latitude, longitude, timestamp, and other information. Wi-Fi is handled in a different way than the previous two location schemes. We gather the Wi-Fi parameters by calling the Wifi Manager Service to get the Wi-Fi access points. We register our Broadcast Receiver Wi-Fi receiver with an Intent Filer action to get back the broadcasted Wi-Fi scanned intent and then request for and consequently process the real scanned access point's data.
In our CBAC policy system, we give users with a utility to describe physical locations by either capturing snapshots of location data of the required areas or by manually entering the area location coordinates.
When the device is stationary or fixed the accumulated data, which consists of Wi-Fi access points with signal strength ranges further to GPS and cellular location data as supporting location information, will represent one physical defined location to the user. We assign a default policy limitations for the user to configure if at all the device is located in an unregistered location.

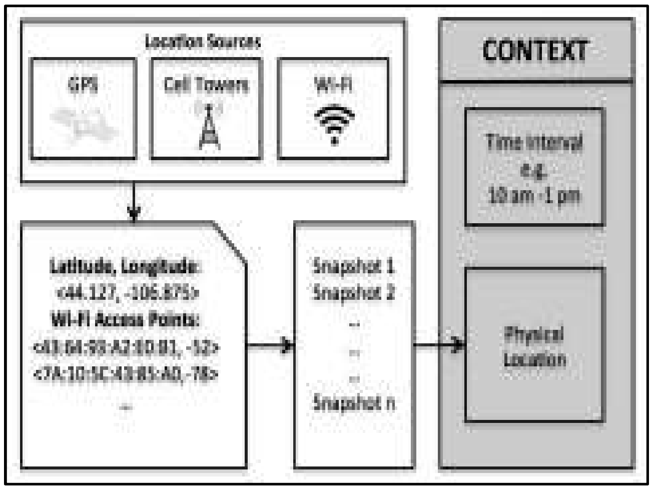

Figure 2: Location Capturing Stage

Location Detection Stage: Figure 3 shows how device context is determined and matched with pre-defined context. Occasionally, the location background service is re-activated to accumulate location-context data to determine the device's current locations. When registering and scanning a sub-area in the location capturing stage, we scan the device's location-related data. The list of user-registered areas that have a subset of the scanned neighboring access points are extracted from the database that exits in mobile. Matching distinct access points is less expensive than determining if coordinate location falls within the limit or not. Then, using the current signal strengths of the access points, we eliminate the list to only a set of best-matched of physical locations whose access points fall within the current captured signal strength parameters.
If the current scanned GPS or cell network coordinates exits within the convex hull of the associated sub-area, then it is likely that the sub-area has been found.

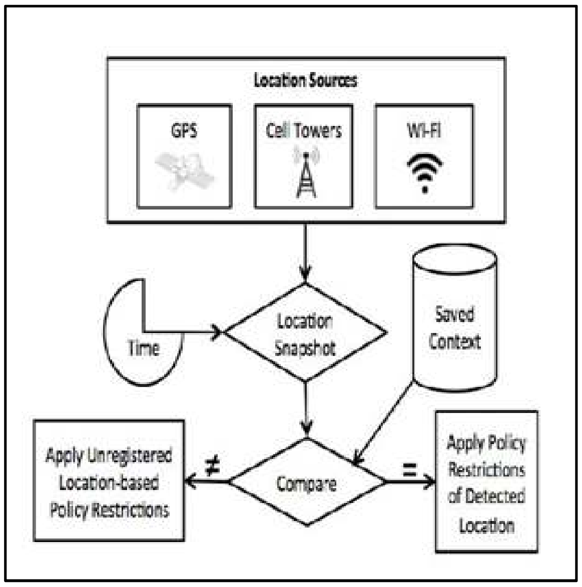
Location Sources
GPS
Cell Towers
Wi-Fi
Time
Location Snapshot
Saved Context
Apply Unregistered Location-based Policy Restrictions
≠
Compare
=
Apply Policy Restrictions of Detected Location

Figure 3: Location Detection Stage

RESULTS AND DISCUSSION

In this part, we declare experimental results about the CBAC mechanism and find its impact on the mobile applications. We added logging commands in various parts of the operating systems where changes were made to observe, such as application access events.

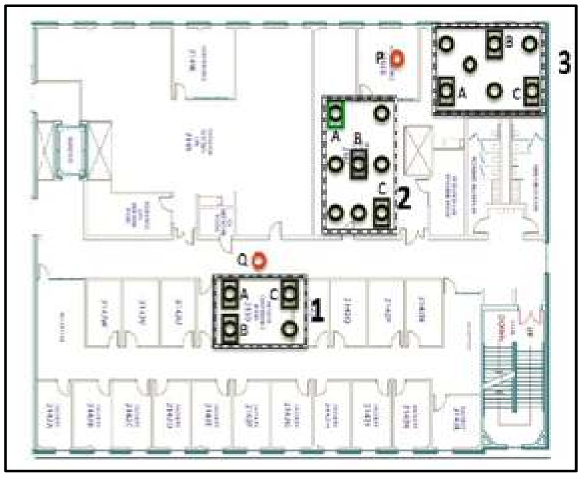

Figure 4: Tested areas in campus buildings

Experiment 1: Location Detection Accuracy. The aim of this experiment is to find the accuracy of the location detection method used in our CBAC mechanism. Figure 4 displays the view of one building where scheme is performed for experiments. The large, grid-pattern rectangles actually refer main locations or areas, indicated by numbers. Areas outside the rectangles are called as "unregistered." The black circles indicate the specific sub-areas examined during the location capturing stage. All other colors indicate other sub-areas analyzed during the detection stage.

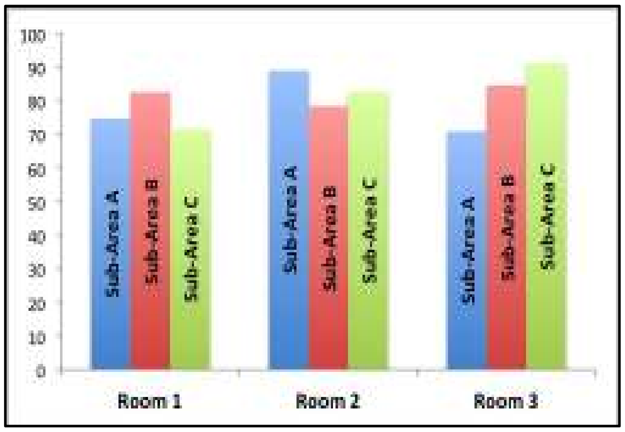

Figure 5: Detection accuracy rate of nearly related areas

Figure 5 displays the detection accuracy in the 3 sub-areas of three rooms such as 1, 2 and 3. At each of the sub-areas of each room either 1, 2 or 3, we performed 50 location tests and measured the number of successful detections. It was supported by testing several "unregistered" areas around the registered rooms.

Experiment 2: Effect of Authorization Limitations: The goal of this experiment is to find the impact of Authorization-related policy limitations on applications. We are interested in whether or not an application crashes as a result of being ignored a Authorization that was at first issued at installation time. Therefore, we performed a stress test on each application and observed the impact on the application upon cancelling its authorizations when requesting a resource. We measured an application as crashing even if it crashed during the execution of one small functionality. The main cause of these crashes is probably due to the developers' mishandling the denial of previously allowed authorizations.
Experiment 3: Performance Overhead. The goal of this experiment is to calculate the timing overhead introduced by our modifications to the Android OS. We compute the amount of time it takes for our changed methods to fully execute once called by applications. We also compare the execution times of these methods before our changes to calculate the overhead introduced by our modifications.

CONCLUSION

In this work, we proposed a changed version of the Android platforms supporting context-based access control policies. These policies limit applications from accessing specific data and resources based on the user specific context. The limitations specified in a policy are repeatedly applied as soon as the user device matches the pre-defined user context related with the policy. Our experimental results show the efficiency of these policies on the Android system and applications, and the correctness in locating the device within a user-defined context. Future work could be to extend our approach to give network administrators of organizations the same capabilities once a moving device connects to their network. By this way, network administrators are given the ability to block malicious application accesses to resources and services that may affect the security of their network.

REFERENCES

1. Templeman R, Rahman Z, Crandall D J, and Kapadia A, Placeraider: Virtual theft in physical spaces with mobile phones, CoRR, 2012.
2. Schlegel R, Zhang K, Zhou X, Intwala M, Kapadia A, and Wang X, Sound comber: A Stealthy and Context-Aware Sound Trojan for Mobile phones, in Proceedings of the 18th Annual Network & Distributed System Security Symposium , 2011.
3. Laboratory L L N, Controlled items that are prohibited on llnlproporty, https://www.llnl.gov/about/controlleditems.html.
4. Conti M, Nguyen V T N, and Crispo B, Crepe: context-related policy enforcement for android, in Proceedings of the 13th international conference on Information security, ser. ISC'10. Berlin, Heidelberg: Springer-Verlag, 2011:331–345.
5. Kushwaha and Kushwaha V, Location based services using android mobile operating system, International Journal of Advances in Engineering and Technology, 2011; 1(1): 14–20.

6. Kumar S, Qadeer M A, and Gupta A, Location based services using android, in Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications, ser. IMSAA'09, 2009:335–339.
7. Kirkpatrick M S and Bertino E, Enforcing spatial constraints for mobile rbac systems, in Proceedings of the 15th ACM symposium on Access control models and technologies, ser. SACMAT '10. New York, NY, USA: ACM, 2010:99–108.
8. Gupta A, Miettinen M, Asokan N, and Nagy M, Intuitive security policy configuration in mobile devices using context profiling, in IEEE International Conference on Social Computing, ser. SOCIALCOM-PASSAT '12. Washington, DC, USA: IEEE Calculater Society, 2012:471–480.
9. Enck W, Ongtang M, and McDaniel P, Understanding android security, Security Privacy, IEEE, 7(1)2009:50–57.
10. Trevisani E and Vitaletti A, Cell-id location technique, limits and benefits: an experimental study, in Mobile Computing Systems and Applications, Sixth IEEE Workshop on, 2004:51–60.

J. Vinothkumar received his Master and Bachelor degree in Computer Science Engineering from affiliated colleges of Anna University, India. His research interests are computer networks, network security and cloud computing.

Vikramarajan Jambulingam received his Master degree in Power Electronics and Drives and Bachelor degree in Electrical and Electronics Engineering from VIT University, India. His research interests are power electronic applications, power quality, power converter and computer networks.

Source of support: Nil, Conflict of interest: None Declared