# CtRBAC:
# Context-Related Role-Based Access Control Mechanism on Android Smartphone

Thiri The` Wut Yee
thirithewutyee@gmail.com
University of Computer Studies, Yangon

Nilar Thein
nilarthein@gmail.com
University of Computer Studies, Yangon

## Abstract

*Smartphones are more and more gaining popularity, creating novel application areas as their capabilities increase in terms of computational power, sensors and communication. Emerging new features of mobile applications and devices give opportunity to new threats. As a result, research addressing information access in smartphones environment has proliferated. The important feature of smartphones is to restrict the behavior of users using applications and services to a certain level and the existing access control mechanism on smartphones mostly holds a coarse-grained.*

*This paper proposes how role-based access control mechanism has motivated the creation of fine-grained access control mechanism. In this paper, a finer access control mechanism which is called context-related role based access control (CtRBAC) is presented. CtRBAC is based on traditional role based access control by incorporating with the contextual information of user and system environment. CtRBAC categorizes the mobile phone users according to their access rights of device's resources and services. By using simple policy and context, the system fulfills necessity of existing access control mechanism.*

Keywords: access control mechanism, Android, context, role-based

## 1. Introduction

In recent years, the revolution of mobile phones era has brought the innovative smartphone technology; providing increased yet complex capabilities. Smartphones are more and more gaining popularity, creating novel application areas as their capabilities increase in terms of computational power, sensors and communication. Emerging new features of mobile applications and devices give opportunity to new threats. As a result, research addressing information access in smartphones environment has proliferated. The important feature of smartphones is to restrict the behavior of users using applications and services to a certain level and the existing access control mechanism on smartphones mostly holds a coarse-grained. In the mean time, most of current systems work on device level security which is per application basis, particularly at installation. User installing third party applications has to trust that the application will not misuse device's resources. Similarly, if user wants to use that application, he must have to grant all permission requests. This all or nothing decision leads to coarse-grained access control mechanism. In addition, as soon as user grants the permissions, there is no way to restrict or revoke these permissions based on user current activities except from uninstalling that application. For example, a user might want to restrict amount of SMS sent for a day to save charge fees by using contextual information such as access patterns. To address these challenges, this paper proposes a fine-grained access control framework for smartphone; context related role based access control mechanism for Android smartphone platform.

By proposing context-related role-based access control mechanism which can be compatible with any smartphone framework, the following contributions are attained. The existing access control mechanism is promoted to a finer mechanism since contextual information is combined with role based access control mechanism. Moreover, the system allows smartphone users to enforce install time as well as runtime policies to overcome security issue which is the most critical concerned with smartphone users. Unlike the existing context control mechanism of smartphone such as silent mode, airplane mode, the proposed system can automatically as well as dynamically change secure modes by communicating with context inference engine which can lead to access control engine which dynamically monitors usage pattern. Furthermore, based on the proposed framework, a prototype with simpler as well as easy to use interface is constructed to specify run time policies on Android phone's resources and services. Finally, by exploiting CtRBAC on every smartphone, we can inevitably protect against privacy and security concerns regardless of any social network media (such as facebook, twitter etc) or any applications or any user (phone owner, guest user, or stranger).

Roadmap: Section 2 describes overview of Android Security. Section 3 describes proposed system model and framework. Section 4 points out the performance evaluation. Section 5 lists some existing works. Finally, section 6 gives some conclusion remarks.

## 2. Overview of Android Security

Developed by the Open Handset Alliance (led by Google), Android is a widely anticipated open source operating system for mobile devices especially for smartphones that provides a base operating system, an application middleware layer, a Java software development kit (SDK), and a collection of system applications. The purpose is to create an open platform for handsets and make mobile applications interoperable crossing venders.

Android software stack includes many libraries, system utilities as well as core applications such as web browser, dialer, calculator, address book, and significant Google applications e.g. Google map etc. The Android SDK provides the tools and APIs necessary to built up applications on Android platform. 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). Each Android application is executed in its own address space and in a separate DVM. Android applications are developed using pre-defined components: activity that represents a user interface; service that executes background processes; broadcast receiver, a mailbox for communication between applications; content provider, to store and share application's data. Application components communicate through messages (intents). Android inter-component communication (ICC) is similar to the inter-process communication (IPC) in Unix-based systems.

Focusing on security, Android combines two levels of enforcement: at Linux system level, and application framework level, At the Linux system level, Android is a multi-process system: each application runs in its own process and address space. It uses sandboxing technique i.e. after installed, each application package is a assigned a unique Linux user ID which remains constant. This technique prevents other applications from intervening in its operation except by required permissions which are explicitly declared. At the application framework level, Android provides control through ICC reference monitor. The reference monitor provides Mandatory Access Control (MAC) enforcement on how applications access the components. To use protected features, application must declare the required permissions in its package manifest definition. For example, if an application needs to monitor incoming SMSs, the AndroidManifest.xml included in the application's package must specify that permission [1].

Permissions declared in the package manifest are granted at the installation time and cannot be modified later. Thus, the current Android security model cannot serve our purpose of enforcing fine-grained context-related access control policies. In fact, there are no mechanisms either to enforce or to change policies at application run-time. Alternatively, after authorization for the application, the system will not ask for more permission again. Thus Android provides coarse-grained security level i.e. neither to enforce or to change security policies at application run-time. In order to restrict access dynamically, it is practical to leverage context-related information. Context aware access control mechanism is a mobile computing paradigm in which applications can discover and take advantage of contextual information such as user location, time zone, nearby people and devices and user activity etc and is exploited in decision making of access control. Context-aware system offers new opportunities for developers and end users by gathering context data and adapting system behavior according to the security policies. In combination with mobile devices, this mechanism is of high value and is used to increase usability. Moreover, context-related access control model enhances Android's security mechanism to fine-grained manner [3].

## 3. Proposed System

This section presents an overview of CtRBAC model. Taking the advantage of RBAC and contextual information, the proposed system model offers user an easy-to-use interface allowing runtime policy specification on smartphone's resources within a minimum overhead. The process flow of the CtRBAC model is illustrated in figure 1.

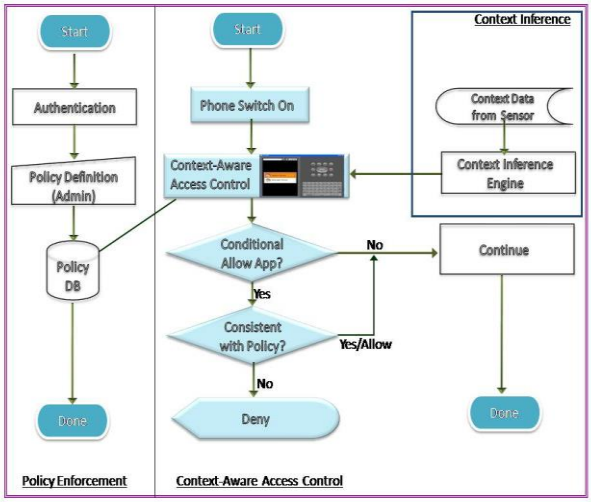


**Figure 1: Process flow diagram of CtRBAC**

The proposed framework consists of three components: context-aware access control (CAAC) engine which controls usage decisions, policy enforcement (PE) component and context inference

(CI) engine which customized captured context information to be suited for decision making process. The following figure shows the overview of proposed system framework.

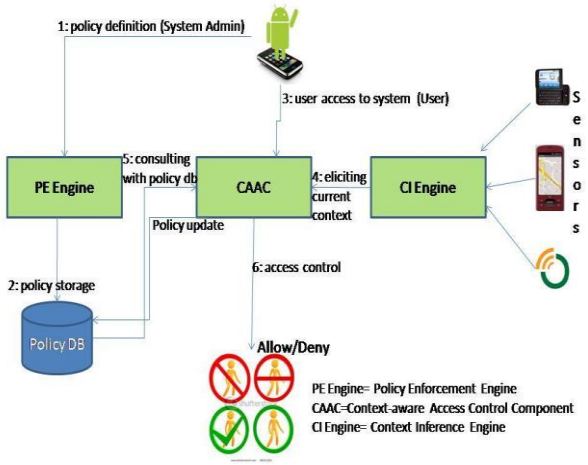


**Figure 2: Overview of proposed system framework**

Firstly, phone owner (system admin) sets access policies via PE engine and these policies are stored in policy database. Thereafter, as soon as phone is switched on, CAAC is instantiated. It prompts the user a home screen with two options (admin home and end-user home). If user chooses end-user mode, system will launch home screen according to the normal user privileges. When user selects admin mode, correspondence screen will be displayed via user authentication. From user point of view, end-user home screen will not differ from that of admin. However, only admin privilege can impose access control policy of phone resources for both user types. The device's environment information i.e. context is monitored by CI and instantly reported to CAAC in decision making process.

## 4. System Evaluation

For practical evaluation, the prototype of proposed system is implemented and its security and overhead are accessed and measured.

### 4.1 Running Example

This section presents how CtRBAC works in practical scenario. From access control aspect, let's consider about parental control point of view. Suppose that parents want to control children's phone usage. Parents do not want their children to spend too much time playing online games or accessing Internet. For example, there is a policy which daddy sets such as "within class hours, child can't access all resources except from parental phone call, educational applications etc". In the mean time, CtRBAC monitors children mobile phone usage and determines whether it is allowable or not by conferring with policies database. If access goes to allowable application or service, CtRBAC passes it. When violated attempt is encountered, CtRBAC immediately inspects current contextual information via context inference engine. If environmental context is matched with policies, access is allowed under conditional exception or totally denied. In such case, there are two situations. When user approaches to predefined or peculiar contextual environment, CtRBAC automatically changes to parental control mode. For instance, mobile phone becomes parental control mode automatically if the temporal context becomes school hours. Otherwise, it becomes parental guardian mode in response with violated access. Predefined policies are treated in priority basis, and the priority must be defined by system admin. It is mostly different from existing mechanism of every mobile phone like silent or airplane mode.

### 4.2 Security

CtRBAC does not reduce the existing Android security. For each access to the device's resources or services, CtRBAC poses further checks i.e. checks according to active policies. Normal accesses are passed on to original Android security check and not influenced by CtRBAC. Android protects applications and data through a combination of two enforcement mechanisms i.e. at system level and at ICC level, These defines the core security framework of existing Android mechanism and the security builds on the guarantees provided by the underlying Linux system. Therefore, the proposed system only reduces allowable access to boost the security not to reduce it. Moreover, CtRBAC helps to prevent security compromise which means that security policies can be defined by the user to limit the device access in particular situations. For instance, user can define policy allowing to use Bluetooth only at home or the office, which are trusted environments. Finally, any malicious attempt which violates the defined policies cannot skip the CtRBAC environment.

### 4.3 Overhead

In this section, time and energy overhead are measured for CtRBAC since both are two main issues of smartphones. According to the proposed framework, two main characteristics are identified which induces overhead. The first is permission check and the second is context resolution. Time overhead is induced by CtRBAC checks. However, the system only checks the particular access which violates the user defined policies. Therefore time overhead is trivial compared to Android permission check. For energy overhead, as expected, the energy consumption is still negligible as long as the proposed system doesn't include client/server

communication for any of decision making process.

Nonetheless, the proposed CtRBAC has reasonable amount of overhead from both the time and energy point of view though overhead increases with increasing the number of rules. Overall, the experimentation of proposed system is still underway and the detail evaluation is left for further study.

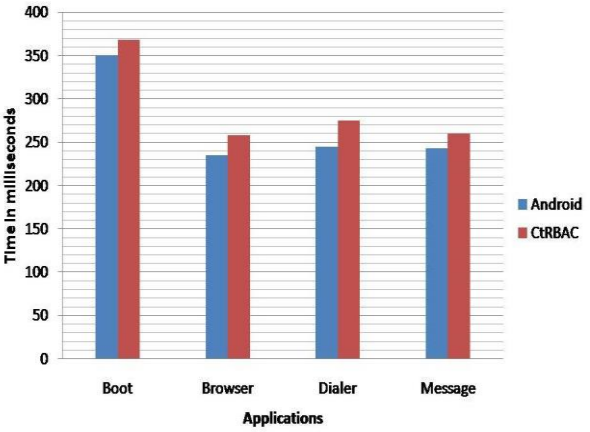


**Figure 3: Average time taken for some applications**

To evaluate the system performance, sample applications such as browser, dialer and message as well as system boot operation are measured in milliseconds. The initializing time for these applications on CtRBAC is measured against the existing Android platform and the results are shown in figure 3. During these measurements, nearly 50% of computations are exploited for CAAC component. Nonetheless, as previously mentioned, the total time overhead is still acceptable for the proposed system.

## 5. Related Work

As the issues described are nontrivial, it has gained intentions from researchers. The research community has been investigated secure solutions for smartphones. Yet, less convincing results have been obtained for enforcing security at application run time. This is because of the limited nature of smartphone in terms of battery and memory overhead.

SCanDroid is a tool for automated security certification of Android applications. It statically analyzes data flows through Android applications, and makes security-relevant decision automatically. It is a reasonable model for offline certification [5]. It only tries to improve the existing Android security mechanism to be best practice. However, it mainly depends on source code inspection and Android manifest file; thus it is not applicable for average user.

Another work concerned about fine-grained access control is Saint [11] in which proposed enhance security mechanism of Android by improving install and run-time policies. Communication between applications or components is subjected to security policies asserted by both the caller and callee applications. In such way, device security is controlled by application provider's policy and not by the user's policy. It's only suitable for developer. Moreover, it can pose only install-time policies.

In addition, the increased number of GNU GPL license applications results in a greater chance of installing Trojans and similar malware. W. Enck et. al. propose Kirin [12] security service for Android, which performs lightweight certification of applications to mitigate malware at install time.

In Paranoid Android [8], an alternative solution is proposed where security checks are applied on remote security servers which host exact replicas of phones in virtual environments. It is a security model that performs attack detection on a remote server in the Cloud where the execution of software on the phone is mirrored in a virtual machine. Although it can deal with security of applications, it mainly depends on the Cloud.

Finally, some researches have been carried out in the area of modeling context aware system to provide meaningful and valuable context information rather than raw context data to the system. In paper [7], ContextDroid is presented and it is designed to provide application developers with the services required to easily build context aware application with an eye towards reducing overhead. Above all, the finer access control mechanism is still needed for smartphone system. Furthermore, the simpler and the more general policy model is welcome in such a way for easy implementation of an access control system regardless of user defined policies. The endeavor of this paper is to fulfill such requirements to some extent.

## 6. Conclusion

Android security mechanism is device level security, works on per application basis, typically at install. Obviously it is coarse-grained mechanism. To protect confidential content and the integrity of services, there should be a framework which dynamically allows and restricts access to resources and services. While this mechanism is achieved by user-centric, context-related security mechanism, the effort is still medium in research area since the security policies are hard to define and learn. Through ongoing study, a valuable yet feasible framework which exploits user context information to provide fine-grained security control mechanism is proposed. In addition, by using simple policy and context model with the aid of clear subject/object mapping mechanism, it can be concluded that the system will fulfill the security needs within minimum performance overhead.

## References


[1] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev "Google Android: A State-of-the-art Review of Security Mechanisms", 2009.

[2] Jesse Burns "Developing secure mobile applications for Android", 2008.

[3] P. Samarati, S. D. di Vimercati, "Access control: policies, models, and mechanisms", Foundations of Security Analysis and Design, Tutorial Lectures, vol 2171, 2001, pages 137-196.

[4] W. Enck, M. Ongtang, P. McDaniel, "Understanding Android security", IEEE Security & Privacy Magazine, Vol. 7(1), January/February 2009, pages 10-17.

[5] A. P. Fuchs, A. Chaudhuri and J. S. Foster. "ScanDroid: automated security certification of Android applications", 2009.

[6] B. Wissen, N. J Palmer, R. Kemp, T. Kielmann, H. Bal, "ContextDroid: an expression-based context framework for Android", 2010.

[7] G. Bai, L. Gu, T. Feng, Y. Guo, X. Chen, "Context-aware usage control for Android", In 6th International ICST Conference on Security and Privacy in Communication Networks (SecureCOMM ), Singapore, September 7-9, 2010.

[8] G. Portokalidis, P. Homburg, K. Anagnostakis, H. Bos, "Paranoid Android: zero-day protection for smart phones using the cloud", Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC '10), 2010, pages 347-356.

[9] M. Naunam, S. Khan, " Apex: extending Android permission model and enforcement with user-defined runtime constraints", Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010, pp. 328-332.

[10] M. Conti, V. T. N Nguyen and B. Crispo , "CRePE: context-related policy enforcement for Android",

[11] M. Ongtang, S. McLaughlin, W. Enck , P. McDaniel, "Semantically rich application-centric security in Android", Proceedings of Annual Computer Security Applications Conference (ACSAC 2009), December 2009.

[12] W. Enck, M. Ongtang, P. McDaniel, "On lightweight mobile phone application certification", Proceedings of the 16th ACM Conference on Computer and Communications Security, November 2009.

[13] X. Ni, Z. Yang, X. Bai, A. C. Champion and D. Xuan, "DiffUser: differentiated user access control on smartphones", Proceedings of International Workshop on Wireless and Sensor Networks Security (WSNS'09) conjunction with IEEE Mobile and Ad-hoc Sensor Systems (MASS), Oct. 2009.

[14] Z. Liu, C. S. Nam and D. R. Shin, "UAMDroid: user authority manager for the Android platform", International Conference on Advanced Communication Technology (ICACT), IEEE Press, Feb 2011, pp. 1146 – 1150.