Activity-Oriented Access Control for Ubiquitous Environments

Le Xuan Hung, Riaz Ahmed Shaikh, Hassan Jameel, S.M.K.R. Raazi, Yuan Weiwei, Ngo Trong Canh, P.T.H. Truc, Sungyoung Lee
Dept. of Computer Engineering, Kyung Hee University, Korea,
{lxhung, riaz,hassan, raazi,weiwei,ntcanh,ptttruc, sylee}@oslab.khu.ac.kr,

Heejo Lee
Dept. of Computer Science and Engineering, Korea University, Korea
heejo@korea.ac.kr

Yuseung Son, Miguel Fernandes
Dept. of Information Assurance, Institute for Graphic Interfaces, Korea
{yssohn, mfernandes}@igi.re.kr

***Abstract*—Recent research on ubiquitous computing has introduced a new concept of activity-based computing as a way of thinking about supporting human activities in ubiquitous computing environment. Existing access control approaches such as RBAC, became inappropriate to support this concept because they do not consider human activities. In this paper, we propose Activity-Oriented Access Control (AOAC) model, aiming to support user's activity in ubiquitous environments. We have designed and implemented our initial AOAC system. We also built up a simple scenario in order to illustrate how it supports user activities. The results have shown that AOAC meets our objectives. Also, AOAC it takes approximately 0.26 second to give a response which proves that AOAC is suitable to work in real-time environments.**

***Index Terms*—access control, ubiquitous computing, activity**

I. Introduction

Recent research on ubiquitous computing [1] has introduced a new concept of Activity-Based Computing (ABC) as a way of thinking about supporting human activities in ubiquitous computing environment [2]. The application-centered and document-centered computing paradigms have proved successful for programming in their respective domains: the application-centered paradigm fits large, centralized, business domains like banking, while the document-centered paradigm supports office-type work. However, it is not clear that these paradigms are the proper ones for programming pervasive computing technology. As a consequence, we have proposed and explored an activity-centered perspective for modeling an important class of pervasive computing systems. Our main thesis is that the computing system must support the handling of human *work activities* directly; similar to how document-centered systems support handling documents directly. "Work activity" means well defined tasks or processes that a person has to carry out as part of his/her job, often using computers as part of the activity. For example, when a project leader is discussing with his team about the project, related project information should be displayed on the wall-based display in the room.

In order to support that thesis, access to ubiquitous computing resources must be controlled in such a way that privacy is guaranteed along with the flexibility to change user access permissions when his/her activity is changed. Traditional access control models exploit user identity/role information to determine the set of access permissions [3][5]-[7][10]. The policy specifications of these models tightly couple identity/role of users with their permissions. This coupling does not support user activities. Other works use context as a foundation to authorize access [9]. However, the concept of context is general, for example location context, time context, system context, etc. It does not precisely specify user activities. As a consequence, these approaches are not appropriate to work in activity-centered environments.

In this paper, we propose a new approach, Activity-Oriented Access Control model (AOAC), aims to support user activities in ubiquitous environments. In AOAC model, each user is allowed to perform certain activities. Each activity is associated with a number of access permissions. As the initial stage, we exploit ubiquitous manufacturing systems as our system environment in order to present and illustrate the proposed model.

The rest of the paper is organized as follows. Section II briefly discusses related work. Section III presents AOAC model. We describe the design and implementation in Section IV. Section V verbally discusses security of the model. Finally, Section VI concludes the paper and outlines our future work.

II. Related Work

Access control technology has a long history. Started in late 60s, Lampson [3] introduced a formal, mathematical description of access control based on an *access matrix*. In 1973, Bell and Lapadula [4] proposed the first *rule-based*

access control model. In 1983, access control technology took a significant step forward when the U.S. Department of Defense (DoD) published its *Trusted Computer System Evaluation Criteria* (TCSEC) [5]. However, one of the big limitations of these models is that they manage privileges based on individuals instead of group of individuals. This brings a high complexity and significant cost to manage in-growing large-scale systems. Therefore, during early 90s, a new access control model based on *role* of user (*Role-based Access Control* - RBAC) was introduced by D. Ferraiolo [6] to tackle this problem. RBAC is conceptually simple: access to computer system objects is based on a user's role in an organization. Roles are authorized instead of users. A role denotes a job function describing the authority and responsibility conferred on a users assigned to that role. RBAC is a basis for many descendant models afterwards.

Convington *et al* [7] proposes a *Generalized Role-Based Access Control* model (GRBAC). GRBAC enhances RBAC by introducing three different kinds of role. The *Usage Control* ($UCON_{ABC}$) model [8] encompasses traditional access control, trust management, and digital rights management (DRM) to control the access to and usage of digital information objects. $UCON_{ABC}$ enables finer-grained control over usage of digital objects than that of traditional access control policies and models. These approaches exploit user identity/role information to determine the set of access permissions. Therefore, they are inappropriate to apply in ubiquitous environments where access permissions are tightly coupled with the activities, rather than user identities or roles

Corradi *et al* proposes a new model of context-based access control, *Ubiquitous Computing Context-based Security Middleware* (UbiCOSM) [9]. UbiCOSM uses the context as a foundation for security policy specification and enforcement processes. Unlike RBAC, permissions are directly associated with contexts, instead of user identities/roles. UbiCOSM is similar to our approach in a sense that it avoids exploiting user/role information to determine the set of user permissions. However, it differs from AOAC because UbiCOSM's context is general (e.g. location, time, etc). This does not directly specify actual activities of users.

III. Activity-Oriented Access Control

We abstract the model in three levels: user level, activity level, and privilege level, as illustrated in Fig. 1. Each user holds a number of credentials [11] specifying his attributes such as role, experience, and assignment. A user is authorized to perform a certain activity if his attributes are satisfied the system policies. Each activity is associated with a number of access privileges. As an example, Alice, who is project leader, assigns Bob to develop a module *M*; since Bob is authorized to carry out 'developing module M', he has a permission to access all materials related to *M* including the source code, technical report, progress, etc. *Assignment* is our term to specify privilege delegation in AOAC. If Alice wants to delegate some rights to a user Bob, Alice activates an *assignment* process in which she assigns *assignment credential* to Bob. An assignment credential may carry on access rights or particular attributes (such as role, qualification, etc.) so that Bob can activate the activity.

A. Assignment: Privilege Delegation via Digital Credentials

Basically, *privilege delegation* is a term to indicate that a user delegates to another user a particular privilege. In AOAC, we model privilege delegation by *assignment*, where U_1 is an *assigner* and U_2 is an *assignee*. *Assignment* is a similar term to *appointment* in [10]. It occurs when a user grants a digital credential that directly or indirectly allows another user to perform one or more activities. Credential's content may be an assignment of activities (*direct delegation*), or may be an assignment of role, qualification, etc so that the user may use to activate some activity (*indirect delegation*). Our delegation approach differs from *appointment* in several aspects. Firstly, Alice may delegate an access right to Bob, and Bob may further delegate this right to Carol, and so on. We call this *multi-step delegation*. Secondly, our privilege delegation may be *restricted* or *unrestricted*. It means that Alice may want to restrict how Bob can further delegate its access right. Delegation in *appointment* approach only considers how to delegate a credential to another user in order to activate one or more roles. It does not concern further delegation or restriction.

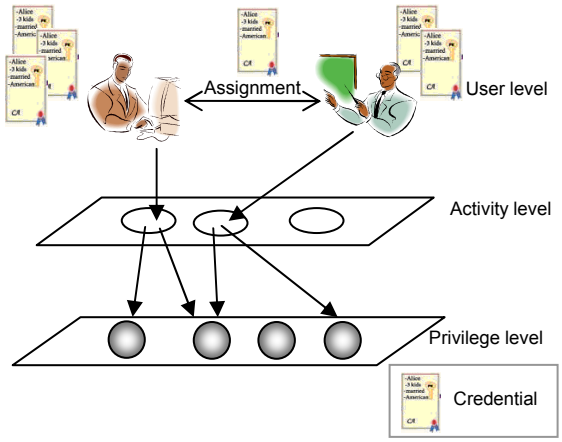

Fig. 1. Abstract levels of AOAC

B. Privilege Revocation

Privilege revocation is very important. There are many situations, in which credentials should be revoked. For example, Alice delegated a credential to Bob for a particular business; if the business is accomplished, or Bob is transferred to another department, the credential should be revoked immediately. There are different reasons and different ways to revoke delegated credentials. Privilege revocation can be done in four ways [10]:

- *By its assigner only*: only assigner can revoke the delegated credential. A credential revocation is actually a new delegation which sets negative effect on the delegated credential.
- *By anyone active in the credential*: there are some undesired situations that the assigner is not possible to revoke his

delegated credential. In this case, a flexile solution is to allow anyone who is specified in the set of credential's assigners to revoke the credential on behalf of the assigner.

- *By assignee's resignation*: the assignee may resign the credential once s/he no longer wants to use it.
- *By rule-based system revocation*: revocations can be carried out by the system itself. There are different ways of revocation: time duration on the credential is expired; constraint on the credential is violated; the task is completed; revoked at the end of the assigner session or the assignee's session.

There are two cases: *single-step revocation* and *multi-step revocation*. Single-step revocations are applied for single-step delegations. It means that Alice only delegates a credential to Bob without any further delegation from Bob. Multi-step invocations are applied for multi-step delegations. However, single-step revocations could be considered as a case of the multi-step revocation with the number of delegation step is one. There are different ways for cascading revocations. However, one of the simplest ways is based on credential identifiers (ids) to revoke them. It means that the original assigner (Alice) attaches a unique id to each credential. Whenever she wants to revoke, she just needs to indicate the credential's ids and the system will consider those credentials as invalid ones.

C. Activity Activation Rules

As aforementioned, a user must hold a number of required attributes so that s/he can activate his/her activity. This is more practical than identity-based approaches which solely rely on user identification such as user name, employee identification. For example, to be granted permission for interviewing a new employee for the company, the user must be a senior, and either belongs to the personnel department or holds an assignment of a personnel department's staff.

We use Prolog-like expression to formulate our *activity activation rule* as follows:

$$ACT \leftarrow ATTR_1, ATTR_2, ..., ATTR_n$$

i.e. an activity *ACT* is allowed to perform if the user holds a set of attributes $\{ATTR_i\}$.

An example of *activity activation rules* is given as follows:

$$employee\_interviewing(X, Y) \leftarrow senior(X), personnel\_dept(X), new\_employee(Y)$$

It is notice that $ATTR_i$ could be any attributes that a user possesses including privileges to perform an activity, privileges to access a resource, etc. It is not restricted to only user's properties or characteristics. User's attributes are encoded in X.509 certificate [11], an ITU-T standard for public key infrastructure (PKI).

D. Permission Activation Rules

Whenever a user activates an activity, corresponding permissions are automatically activated if a number of context constraints are satisfied. We define context constraints as follows:

Definition 2.1: *Context constraints* are defined as any requirement about contextual information such as time, location, etc. Context constraints play a key role in specifying context-sensitive policies. In organizations, it is very important to restrict user's access and invoke user's privileges if contextual requirements are not met.

Permission activation rules are as follows:

$$PERM_1, PERM_2, PERM_3, ..., PERM_n \leftarrow ACT, CC_1,$$

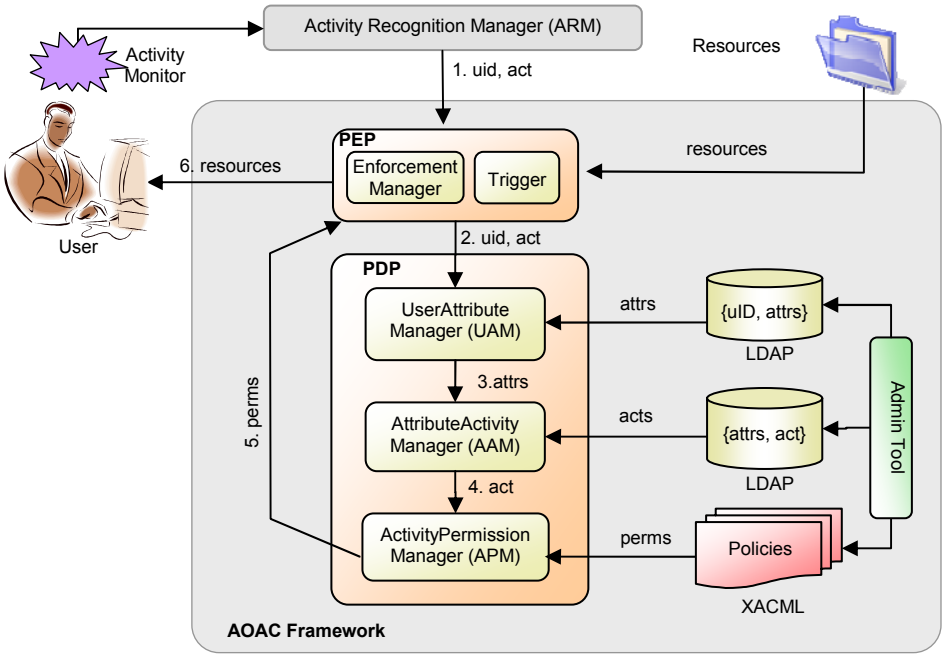

Fig. 2. AOAC System Design

$CC_2, ..., CC_m$

i.e. if user can activate an activity ACT under satisfied context constraints CC_1, CC_2,…,CC_m, then s/he is granted corresponding permissions $PERM_1$, $PERM_2$, $PERM_3$, ..., $PERM_n$.

An example of permission activation rules is:

read (X, employee_profile(Y)) ├ *employee_interviewing(X, Y), within(May, 2008)*

i.e. X can access the profile of the new employee Y within May of 2008 during the company recruiting process if X is allowed to carry out interview activity

IV. System Design and Implementation

We have designed and implemented our initial AOAC system. We have also built up a simple scenario in order to illustrate how it supports user activities in ubiquitous environments. The scenario is described as follows:

Alice, who is the project leader, assigns Bob to develop the access control module. In the meanwhile, Carol, the manager of personnel department, asks Bob to interview a new employee to support Bob in the project. As a consequence, Bob holds credentials specifying his assignment of developing access control module, and interviewing a new employee. He also holds another credential saying that he is a senior of the security team.

At eight o'clock in the morning, Bob comes to the company. He goes directly to the conference room where a new applicant is waiting for him. The system recognizes Bob and knows that he is interviewing the applicant, so the applicant's resume and related information are shown up on his PDA. After one hour, he finishes the interview and leaves the room. He enters his room, turns on his computer and starts developing his module. The monitor displays the program code, project architecture, and some new project-related tasks that he has to accomplish today.

User's attributes, activities and activity activation rules are stored on a Lightweight Directory Access Protocol (LDAP) server. Permission activation rules are defined by eXtensible Access Control Markup Language (XACML) standard [12]. The system design is depicted in Fig 2. To be compliant with XACML, AOAC includes a Policy Enforcement Point (PEP), and a Policy Decision Point (PDP). PEP performs access control by making decision of requests and enforcing authorization decisions. PDP evaluates applicable policy and renders and authorization decision. The PDP is composed of three sub-components: User-Attribute Manager (UAM), which retrieves user's attributes according to user identifier (uid) from user's attributes LDAP server; Attribute-Activity Manager (AAM), which matches user's attributes to a set of allowed activities; and Activity-Permission Manager (APM), which retrieves all access privileges for given activities from XACML policies. The administration tool is used by the system administrator to define activities and policies. The system operates by the following steps:

1. ARM provides user's activity information to AOAC by gathering raw contextual data related to user activity, producing high level context, and then reasoning user activity.
2. PEP forwards user id (*uid*) and current activity (*act*) to UAM.
3. UAM queries all attributes (*attrs*) matched to *uid* from LDAP server. It then sends those attributes to AAM.
4. According to user's attributes, AAM looks up LDAP server and makes decision if the user is allowed to perform the given activity (*act*). If yes, AAM forwards the activity (*act*) to APM.
5. At APM, corresponding permissions (*perms*) for the activity are achieved by checking the system policies. PDP then sends list of permissions to PEP.
6. At PEP, the Trigger module is invoked to retrieve resources. Then the resources are sent to the user's device.

Our implementation and demonstration of the above scenario have shown that AOAC works well with quick response time to user. It takes approximately 0.26 second to display corresponding resources on the user's device after recognizing a user activity. This shows that AOAC can operate in real-time and in a flexible manner in ubiquitous environments.

V. Discussion

In this section, we verbally discuss possible vulnerabilities of the proposed scheme. The discussion is focused on the confidentiality and integrity since these are two critical preservation of access control.

A malicious user manipulates context data in order to fool the system. However, in AOAC context data provides user activities to trigger PEP (see Fig. 2). It is not a requirement to authorize information and services to users. Therefore, if the context data is incorrect, then the user would not get the corresponding information and services according to his activity. It does not violate confidentiality or integrity of information. Another threat is that a malicious user can forge credentials to access the system. However, credential security is beyond the scope of this paper. It depends on the security level of PKI. For AOAC, as long as the user credentials are secure, unauthorized users would not be able to access the system. In the proposed access control scheme, a user is not able to access to the system if he does not hold a number of required credentials. Therefore, unauthorized access is not possible. As consequence, confidentiality and integrity of information are preserved.

VI. Conclusion and Future Work

Motivated from Bardram *et al*'s work of activity-based computing, we propose an Activity-Oriented Access Control (AOAC) model aiming to support user's activities in ubiquitous environments. Center of AOAC is user's activity. Each user is allowed to perform a number of activities. Each activity is associated with a set of access privileges so that the user can accomplish the activity.

In this study, we have introduced AOAC model and the system design. We also implemented and demonstrated a simple scenario in order to show how AOAC supports user's activity in ubiquitous manufacturing systems. Several issues are still unresolved. One of those is how to recognize user's activity as discussed in [13]. We chose the first solution**,** in which the users explicitly state their activity. As our future work, we will focus more on activity recognition. We will also extend our scenario into ubiquitous hospitals and other ubiquitous environments.

Acknowledgements

This work was supported by the IT R&D program of MIC (Ministry of Information and Communication) / IITA (Institute of Information Technology Assessment). [2005-S-604-02, Realistic Virtual Engineering Technology Development].

References

[1] M. Weiser: Scientifc America. The Computer for the 21st Century. (Sept. 1991) 94-104; reprinted in IEEE Pervasive Computing. (Mar. 2002) 19-25.

[2] E. Bardram. Activity-based computing: support for mobility and collaboration in ubiquitous computing. Personal and Ubiquitous Computing, vol.9(5), pp.312-322, September 2005.

[3] Lampson, B. W., "Dynamic Protection Structures" AFIPS Conference Proc, 35, 1969, pp. 27–38

[4] Bell, D. E., and L. J. LaPadula, Secure Computer Systems: Mathematical Foundations and Model, Bedford, MA: The Mitre Corporation, 1973

[5] DoD Trusted Computer System Evaluation Criteria (TCSEC) , DoD 5200.28-STD.

[6] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transaction on Information and System Security, Vol. 4, No. 3, August 2001, pages 224-274

[7] M. J. Covington, M. J. Moyer, and M. Ahamad, "Generalized Role-Based Access Control for Securing Future Applications," 23rd National Information Systems Security Conference, 2000.

[8] J. Park and R. Sandhu. The UCONABC usage control model. ACM Transactions on Information and System Security (TISSEC). Volume: 7 Issue: 1 p. 128 – 174. 2004.

[9] A. Corradi, R. Montanari, and D. Tibaldi, "Context-based access control management in ubiquitous environments," Proc. Third IEEE International Symposium on Network Computing and Applications, (NCA) pp.253–260, Aug. 2004.

[10] Jean Bacon, Ken Moody, Walt Yao. "A model of OASIS role-based access control and its support for active security". ACM Transactions on Information and System Security (TISSEC), Volume 5 Issue 4, 2002

[11] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure: Certificate Management Protocols", RFC 2510, March 1999

[12] XAMCL and OASIS Security Services Technical Committee. eXtendible Access Control Markup Language (XACML) committee specification 2.0. Feb 2005

[13] Jakob E. Bardram. Activity-Based Computing - Lessons Learned and Open Issues. Workshop on Activity - From a Theoretical to a Computational Construct (ECSCW) 2005.