Adapting the UCON_{ABC} Usage Control Policies on CORBASec Infrastructure[1]

Lau Cheuk Lung, Marcelo Shinji Higashiyama, Rafael R. Obelheiro, Joni da Silva Fraga
Graduate Program in Applied Computer Science - PPGIA
Exact Sciences and Technology Center - CCET
Pontifical Catholic University of Paraná (PUCPR) - Curitiba - Paraná - Brazil
lau@ppgia.pucpr.br, shinji@ppgia.pucpr.br, rro@das.ufsc.br, fraga@das.ufsc.br

Abstract

The JaCoWeb-ABC infrastructure is an extension of the CORBASec specification that applies the UCON_{ABC} access control model to its security layer. JaCoWeb-ABC defines configurable access controls that deploy authorization, obligation and condition policies. These security policies can be defined in two different manners. The first one is totally transparent to applications, for cases where JaCoWeb-ABC has all the necessary information for the access decision process, and the second one works together with applications, in cases where security controls depend on external information that must be supplied by the application. Combining these two functionalities allows for a much more accurate and strict control over the actions of users within a system, making it possible to block access in case inappropriate behavior is identified.

1. Introduction

The popularization of the Internet made it possible for any company to use this network as a new communication channel for their business, offering to their clients virtual services, such as e-commerce, Internet banking, auctions, etc. For this reason, companies began to steer systems development towards client-server applications and to look for solutions based on middleware, such as CORBA, RMI, DCOM, and Web Services, that work based on client-initiated transactions that are executed by a server. These services have increasingly become victims of inappropriate behavior by Internet users. In the beginning, some of these malicious users intended only to make these services unavailable (DoS – *Denial of Service*). But more recently, this practice has been moving towards digital crimes, leading to huge losses to companies. This problem has prompted companies to increase their investments in the security of their computing systems.

Thus, many security concepts started being studied and applied to computing systems, like access control models such as the mandatory (MAC) [1], discretionary (DAC) [2], role-based (RBAC) [3], and others. Technologies such as SSL and digital certificates also became widely used, in particular for the purpose of ensuring information confidentiality, integrity and authenticity. Even so, employing these technologies does not always prevent users from carrying abusive or even illegal procedures, if there are no specific security policies to this end.

Focusing on this need, Ravi Sandhu and Jae Hong Park presented in [4], [5], [6] a new access (usage) control model, called UCON_{ABC} (Usage Control), which defines Authorization, oBligation and Condition policies that can be applied dynamically, during the access to an object (e.g. a system resource). In this model, subjects and objects have control attributes that can be altered during the access of a subject to an object (mutability of attributes). An important advantage of this model is its capability for adapting a range of existing access control models to its security policies. In other words, UCON_{ABC} is a framework that can be used to emulate any traditional control access model and also allow mutability of attributes. This way, UCON_{ABC} allows a user to be dynamically controlled for the whole duration of her access to the system – resulting in this way an usage control more precise over all resources of the system.

The CORBA Security Service specification (CORBASec [8]) does not have a specification for access (or usage) control in its security layer. Actually, it provides only a definition of a discretionary model

[1] This work is partially supported by CNPq (Brazilian National Research Council) through processes 481523/2004-9, 506639/2004-5 and 550114/2005-0

which is not enough to fully control the actions of users over system resources. To fill this gap, some proposals aimed to integrate traditional access control models to the CORBA architecture [9, 10, 13, 14]. We believe that a usage control model can be advantageous for many CORBA distributed applications, such as workflow and groupware. In this paper, we propose the JaCoWeb-ABC model that implements usage control mechanisms in the CORBASec platform. Thus, this paper proposes to apply the UCON_{ABC} model in the CORBASec infrastructure, so defining a stricter access control mechanism. This makes it possible to revoke access privileges, for instance, in cases where the system identifies a user's actions as being improper. The main challenge of this proposal is to define a specification capable of implementing the UCON_{ABC} security policies in the CORBASec infrastructure in such a way that authorization, obligation, and condition policies can be dynamically verified.

This paper is organized as follows: section 2 presents traditional access control models. Sections 3 and 4 provide overviews of the UCON_{ABC} model and the CORBASec specification, respectively. Section 5 presents the JaCoWeb-ABC model. Section 6 reviews related work. Finally, section 7 concludes the paper.

2. Traditional access control

Traditional access control models [4], such as discretionary (DAC) [2], mandatory (MAC) [1], and role-based (RBAC) [3], boil down to a Subject (S) trying to access a given Object (O). Permissions are determined based on attributes that characterize the subject (ATT(S)) and the object (ATT(O)), defining the corresponding usage rights (R), see Figure 1.

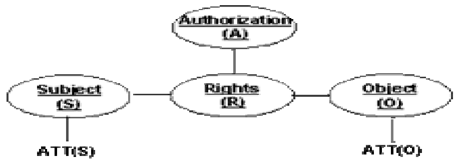

Figure 1. Traditional Access Control

These access control models are based on a static authorization matrix, that is, once a subject is granted access rights to a given object, other situations are no longer taken into account, such as the order in which objects are accessed, controls for abusive usage, or allowing access to objects only after certain conditions and obligations to the system are met. These controls are, when they become really necessary, usually implemented and managed within the applications themselves, and are not always coded in a standardized manner, coherent and free of possible implementation errors (bugs) introduced by system developers.

3. The UCON_{ABC} usage control model

In the UCON_{ABC} usage control model, proposed by Jaehong Park and Ravi Sandhu in [4], [5], [6], policies are defined for *Authorizations*, *oBligations,* and *Conditions* (ABC), making it possible to adapt different access control models inside a single model, resulting in a much more accurate and strict control of the actions granted to a subject in relation to an object. One important feature is that the access rights granted to a subject for an object are determined at run time, and so can be changed according to the actions performed by the user within the system.

The UCON_{ABC} model comprises eight components: *Subjects* (S), *Subject Attributes* (ATT(S)), *Objects* (O), *Object Attributes* (ATT(O)), *Rights* (R), *Authorizations* (A), *oBligations* (B), *Conditions* (C), as shown in Figure 2.

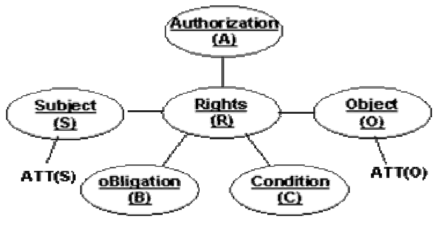

Figure 2. UCONABC Access Control Model

The concepts of *Subject* and *Object* are the same as in traditional access control models. Rights qualify a subject for accessing an object, in a specific mode, such as reading or writing. The main features of the UCON_{ABC} model are related to the changeability/mutability of attributes, wherein the right is not assessed against an existing static access matrix. In this process, the decision is made at run time, at the moment the subject is accessing the object. The ABC assessment processes are defined below:

- **Authorizations**: requirements that must be satisfied before (*preA*) or during (*onA*) permission for a Subject to access an Object. Authorizations are based on subject and object attributes, which specify the given rights. For instance, only a group of users of the system can read a certain file;

- **oBligations**: requirements that the subject must address before (*preB*) or during (*onB*) the access to a given object. Once a subject addresses and meets his system obligations, access to further objects may be granted. For instance, providing an e-mail address before being able to download a document from a company's website;
- **Conditions**: factors related to the system environment, which allow checking of conditions before (*preC*) or during (*onC*) access is granted to a subject for a given object. For instance, to allow a subject access to a specific object only during regular working hours.

4. CORBASec Specification

The CORBA standard, according to the definition of the *Object Management Group* (OMG), specifies a software architecture that supports distributed applications and ensures interoperability across different hardware platforms and operating systems. The CORBA 1.1 specification defined an IDL (*Interface Definition Language*) and an API (*Application Programming Interface*) that allows client-server interaction through an ORB (*Object Request Broker*). The CORBA 2.0 [7] specification underwent a major review process, where several new features were added, among these, the CORBA security service specification (CORBASec [8]).

4.1. CORBA Security Services Specification

The CORBA security model (CORBASec) is an open specification for security in distributed object systems [8, 7]. CORBAsec relates objects and components in four system stratification levels: *the application level*; the *middleware level* formed by object services (COSS), ORB services and the ORB core; the *security technology level* formed by the underlying security services; and finally, the *basic protection level* comprising operating system and hardware functionalities.

The CORBASec specification works with security services at interceptor level, making it possible to protect objects in a manner that is transparent to the application. It is composed of object services (COSS), such as *PrincipalAuthenticator*, *Credential*, *AccessPolicy*, *RequiredRights*, *AccessDecision*, *SecurityManager*, *PolicyCurrent*, *Vault* and *SecurityContext* objects. The *PrincipalAuthenticator* object is responsible for principal authentication: its function is to acquire credentials that will be used to identify a principal (i.e., a subject) in the system. In CORBASec, policies are described through the *security attributes* of system resources (control attributes) and of principals (privilege attributes). CORBASec defines only the *corba* family that contains four types of rights — *g (get)*, *s (set)*, *m (manage)*, and *u (use)* — despite allowing freedom for definition of other families of rights.

5. The JaCoWeb-ABC proposal

UCON$_{ABC}$ defines a very complete model, allowing a range of different access control concepts to be adapted to its functionality. The CORBASec specification is well known and widely disseminated in the industrial and academic environments, integrating a variety of security concepts, such as processes for authentication, authorization, audit, confidentiality, and integrity, but its access control layer define only a simple discretionary model that is not always sufficient to meet the security needs of CORBA-based applications. Thus, we propose to integrate the UCON$_{ABC}$ model to the CORBASec specification, adding these functionalities to its authorization layer, more precisely by extending the *AccessDecision* object.

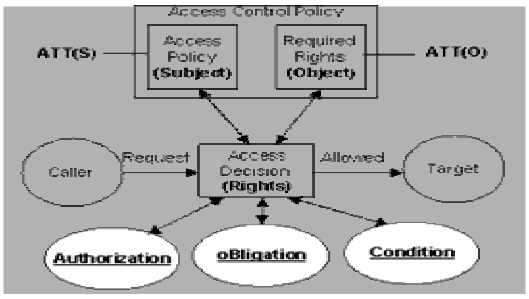

Figure 3 - Analogy ABC x CORBASec.

5.1. Analogy between the UCON$_{ABC}$ and CORBASec models

In order to integrate the UCON$_{ABC}$ and CORBASec models, it is necessary first to identify their components and create an analogy between the models, so that an implementation becomes possible. In Figure 3, we have identified that the *Subject* that represents the client application can be associated to the *AccessPolicy* object, the *Object* represents the invoked methods and can be associated to the *RequiredRights* object, and the *Rights* that represents the access rights can be represented by the *AccessDecision* object, which is responsible for the processes of authorization, obligation, condition, and also mutability of attributes.

5.2. AccessDecisionABC Model

Our proposal for restructuring the CORBASec model in order to adapt it to the $\mathrm{UCON_{ABC}}$ model is introduced in Figure 4. In the conventional CORBASec model, there are only the *AccessDecision*, *RequiredRights* and *AccessPolicy* objects. For this proposal, a model was defined which is capable of controlling the attributes of both subjects and objects, authorization, obligation and condition policies, as well as mutability of object attributes. In order to allow the Subjects and Objects of the $\mathrm{UCON_{ABC}}$ model to have a range of attributes, it was necessary to create the *AttributesManager* object, responsible for including and managing these attributes. In this way, the control over every attribute of the *RequiredRights* and *AccessPolicy* objects, except their identification attributes, was transferred to the *AttributesManager* object. In *Subjects*, an attribute called *Obligations* was created, which stores a history of every obligation ever met by the subject. In *Objects*, one more attribute was defined to contain all the policies of the $\mathrm{UCON_{ABC}}$ model associated to each object (method invoked in the CORBASec model), such as the authorization, obligation and condition processes, in addition to the attribute update policy.

In an access decision process started when a subject begins to access an object, the *AccessDecisionABC* object identifies who is the subject through its credentials that are in the *Current* object, which holds the user credentials, and identifies, through the invoked method, which is the object of this interaction. The $\mathrm{UCON_{ABC}}$ security policies are referenced in the object itself, which holds the definitions for authorization, obligation and condition, and also the policies for mutability of attributes, which can be done before or after the assessment process.

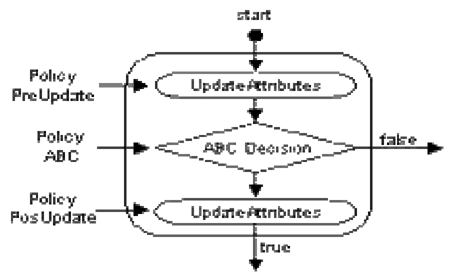

Figure 5. AccessDecisionABC object.

5.3. ABC policy evaluation process

In order to adapt the access decision process and the attribute update process of the $\mathrm{UCON_{ABC}}$ model (defined in Figure 3) to the *AccessDecision* object of CORBASec, we created the *AccessDecisionABC* object, shown in Figure 5.

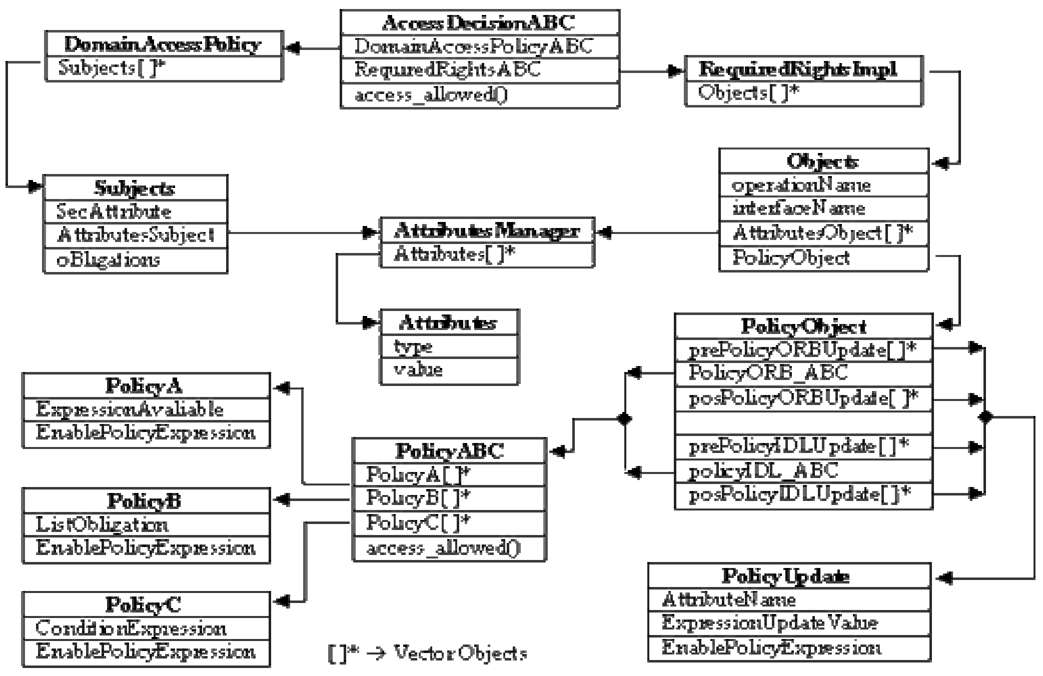

Figure 4 - Components of the AccessDecisionABC Model.

The *AccessDecisionABC* object allows the ABC assessment process to be performed together with the pre-update and post-update processes.

In order to integrate both models (CORBASec and $UCON_{ABC}$), we identified the need to adapt the use of the *AccessDecisionABC* object in two different points. Many applications are security-unaware, and thus can rely on the ORB to transparently perform security policy evaluation. In other cases, however, $UCON_{ABC}$ policies depend on specific application-supplied data; in such cases, the application must cooperate with the security infrastructure, providing the necessary data to the *AccessDecisionABC* object and granting or denying access according to the result (*true* or *false*) of policy evaluation. Figure 6 presents the separation of these activities, creating two extensions to the *AccessDecisionABC* object:

AccessDecisionABC_ORB: ABC access control at middleware level, operating transparently for the application, which requires configuring the security policies that will be controlled by the object. In this case, the decision process depends only on subject and object attributes;

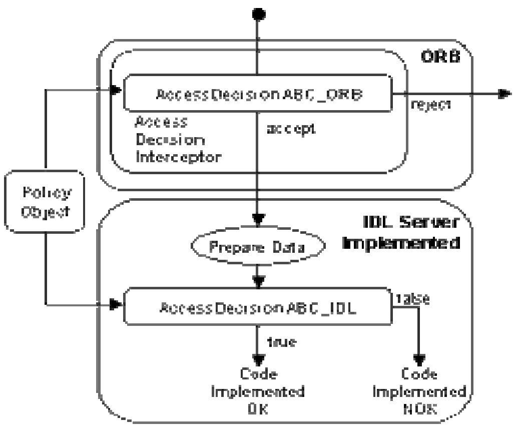

Figure 6. AccessDecisionABC Object

AccessDecisionABC_IDL: ABC access control at application level, with application and infrastructure cooperating in the decision process. In order for this to be possible, the JaCoWeb-ABC architecture provides an API that must be used during the implementation of the IDL (server object), so that it will be able to work together with its authorization layer. This API, called *access_allowed*, must receive four parameters: *Current*, which holds the subject credentials; *ObjectImplemented*, is the name of the object interface; *nameMethod*, is the name of the invoked method; and *Vector*. Vector is an object that allows adding a list of variables that will be used in the policies defined by JaCoWeb-ABC. In this situation, the IDL (Server) must capture external information to this object, such as, for instance, parameters received by the invoked method, prepare these data items to be added into *Vector*, make the call to *access_allowed()* and treat its return value, *true* (access granted) or *false* (access denied), thus continuing with the decision process. The ABC policies can refer to this parameters specifying the reserved word "*parm*[<*index*>]", where *index* means the sequence number of the object added into the vector, starting in 1.

5.4. Mutable attributes

As with the 16 basic models of the ABC family, we have also defined a table containing the six models ($abcORB_1$, $abcORB_0$, $abcORB_3$, $abcIDL_1$, $abcIDL_0$, $abcIDL_3$) for $UCON_{ABC}$ in CORBASec (table 1).

	(0)	(1)	(2)	(3)
abcORB	Y	Y	N/Y	Y
abcIDL	Y	Y	N/Y	Y

(0) – Immutable, (1) – preUpdate,
(2) – onUpdate, (3) – postUpdate

Table 1. JaCoWeb-ABC family

In this table, we have unified the entire ABC assessment process (Figure 5) in two distinct points: abcORB and abcIDL (see Figure 6). Since the CORBA communication model is based on method invocation, the abstraction of ongoing assessment can not be applied directly, as communication starts and ends in the same invocation. However, this abstraction can be applied in CORBA when an application needs a set of invocations to be completed, for instance, a transaction. This way, the first invocation can be considered a pre-access (*preA*, *preB* and *preC*), and subsequent invocations can be seen as ongoing access decision processes (*onA*, *onB* and *onC*).

6. Related work

There are some papers that discuss the implementation of different access control models in CORBASec. In [10], an RBAC model is implemented, allowing the assignment of one or more roles to a principal, which are activated in accordance with the access requirements.

The Resource Access Decision Facility (RAD) [11] is an access control model specified by the OMG. It defines access administration and control policies at application level, which is required when the decision process must be based on external parameters or information that cannot be intercepted by the *AccessDecision* object during the invocation method, and that take business logic into account.

An important related paper that comes close to the UCON_{ABC} model is the Ponder language [12], which uses a declarative language based on objects to define access control policies for authorization and obligation, providing a simple interface for the specification of abstract policies. Ponder authorization policies can be implemented on a range of access control mechanisms such as operating systems, firewalls, and databases.

Since UCON_{ABC} is quite a recent model, we have not found yet any references in the academic community about implementations of the UCON_{ABC} model in applications. However, from the potential displayed, we believe that it will gain widespread adoption as a reference for the implementation of a higher level of access control.

7. Conclusion

The UCON_{ABC} model is a new generation in access control that allows stricter control of the usage made by users of the objects in a system. The JaCoWeb-ABC model introduced significant changes in relation to the latest CORBASec specifications published by OMG [8]. No changes were made to the client-side implementation of CORBA objects as well as issues related to the administration of the security policies.

JaCoWeb-ABC defines a model capable of segregating the activities carried out by an access control layer, in such a way that certain accesses could be controlled in a way that is transparent to the application (abcORB). In other cases, an application will have to work together with JaCoWeb-ABC (abcIDL), due to the access decision for certain objects depending on external information that must be supplied by the application to the JaCoWeb-ABC layer (application-level). The combination of these concepts allows systems to have more accurate and strict control over the actions of users, making it possible to separate the access control logic from the business logic, simplifying the implementation of applications and reducing the possibility of introducing security-related bugs in them.

Since CORBA relies on remote method invocation, we believe that the same solutions proposed in this paper can also be used in the security layers of similar technologies, such as RMI, DCOM and WebServices, in addition to web servers, such as IBM WebSphere or Microsoft IIS. For these implementations to be possible, these technologies need to be analyzed, in order to identify and associate their components to the UCON_{ABC} model.

References

[1] R. S. Sandhu. Lattice-based access control models. IEEE Computer, 26(11) 9-19, November 1993.

[2] D. D. Downs, J. R. Rub, K. C. Kung, and C. S. Jordan. Issues in discretionary access control. In IEEE Symposium on Security and Privacy, pp. 208-218, April 1985.

[3] R. S. Sandhu. Role-Based Access Control. Advances in Computer Science, vol. 46, Academic Press, 1998.

[4] J. Park and R. S. Sandhu. The UCONABC usage control model. ACM Transactions on Information and System Security (TISSEC), Feb. 2004

[5] R. S. Sandhu and J. Park. Usage control: A vision for next generation access control. In 2nd Int. Workshop Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS), 2003, St. Petersburg, Russia.

[6] X. Zhang, J. Park, F. Parisi-Presicce, and R. S. Sandhu. A logical specification for usage control. In 9th ACM Symp on Access Control Models and Technologies, 2004.

[7] Object Management Group. "The Common Object Request Broker 2.0/IIOP Specification", Revision 2.0, OMG Document 96-08-04, 1996.

[8] OMG. Security Service Specification, v1.8. OMG Doc. 02-03-11, Mar. 2002

[9] C. M. Westphall and J. S. Fraga. A large-scale system authorization scheme proposal integrating Java, CORBA and Web security models and a discretionary prototype. In Proc. IEEE LANOMS'99, pp. 14–25, December 1999.

[10] R. R. Obelheiro and J. S. Fraga. Role-based access control for CORBA distributed object systems In 7th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'2002), 2002, San Diego, CA.

[11] OMG. Resource Access Decision Facility, v1.0. OMG Doc. 99-03-02, Mar. 1999

[12] Damianou et al. 2001. The Ponder Policy Specification Language. In Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy'2001).

[13] K. Beznosov, Y. Deng., B. Blakley., C. Burt, and J. Barkley. A resource access decision service for CORBA-based distributed systems. In 15th Annual Computer Security Applications Conf, Phoenix, Arizona, USA. 1999.

[14] K. Beznosov, L. Espinal, and Y. Deng. Performance considerations for CORBA-based application authorization service. In PODC Middleware Symposium, 2000.