Designing Flexible Access Control Models for the Cloud

Salim Khamadja
Laboratoire Systèmes Informatiques
Ecole Militaire Polytechnique
BP-17, Bordj El Bahri, 16111, Alger, Algérie
salim.khamadja@gmail.com

Kamel Adi
Laboratoire de Recherche en Sécurité Informatique (LRSI)
Université du Québec en Outaouais
Gatineau, QC, Canada
kamel.adi@uqo.ca

Luigi Logrippo
Laboratoire de Recherche en Sécurité Informatique (LRSI)
Université du Québec en Outaouais
Gatineau, QC, Canada
luigi.logrippo@uqo.ca

ABSTRACT

In Cloud environments, Cloud users have the possibility to put their sensitive data on Cloud servers, which opens the door to security challenges concerning data protection. In this context, access control is of vital importance, since it provides security mechanisms to protect against inappropriate access to data. Unfortunately, classical access control models such as DAC, MAC, RBAC or ABAC are not sufficiently expressive for highly flexible and dynamic environments such as those found in the Cloud. Often, a combination of elements of these models is necessary in order to properly express varied data protection needs. In this paper, we present a new approach called CatBAC (*Category Based Access Control*), for building dedicated access control models starting from an abstract meta-model. Hence, in our approach, a meta-model can be refined in accordance with the high level security policies of each specific user. Our framework for building access control models can be implemented as a Cloud service and Cloud providers will then apply different concrete access control models produced by each user to process its incoming access requests.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*

General Terms

Security.

Keywords

CatBAC, Meta-model, Access Control Models, Refinement, Hybrid Policies, Cloud Computing

SIN'13, November 26 - 28 2013, Aksaray, Turkey
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2498-4/13/11...$15.00.
http://dx.doi.org/10.1145/2523514.2527005

1. INTRODUCTION

Cloud computing is a concept in the area of "utility computing", offering a service delivery model integrating different dimensions of computing capacity, together with an administrative infrastructure proposed to clients. Among the different definitions of the term "Cloud computing", we retain the one given by Buyya et al. [7] who define a Cloud as a "type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers". This ambitious approach, considering at the same time several forms of deployment and a wide spectrum of features, emphasizes the use of information technology decorrelating the applications and other infrastructure resources underlying its mechanism of distribution. The Cloud delivery models are based on the *Software-Platform-Infrastructure (SPI) framework*. This acronym represents the three major services that can be provided through the Cloud: Software-as-a-service (SaaS); Platform-as-a-service (PaaS); and Infrastructure-as-a-service (IaaS) [12]. More specifically, the SaaS offers to consumers the software that is used under a usage-based pricing model. The PaaS provides the platform for application development, and finally, the IaaS manages the provision of hardware, software and equipment necessary to provide a resource usage-based pricing model.

This service oriented context requires that Cloud providers offer security tools that are sufficiently flexible to meet different needs coming from a wide spectrum of users. When data and systems are hosted in shared hosting environments, access control to data, data privacy and data separation become crucial [15]. Hence, access control can play an important part for data protection needs in Cloud environments. Our paper relates to the newer concept of "Security as a Service", where security services are provided to Cloud users [8, 16]. More specifically, we develop the concept of "Access Control as a Service".

The main goal of access control is to regulate the different operations that may be performed by subjects on objects. Access control models provide a formal representation of access control policies and are the result of a constant evolution in security requirements. Hence, over the last four decades, various approaches to access control have been developed, among which:

- Discretionary access control (DAC) [14] where the primary responsibility for access control belongs to the users who own the information.

- Mandatory access control (MAC) [14] where access control permissions are determined by mandatory hierarchies of data and users.
- Role based access control (RBAC) approach [10], where access control permissions are determined by the role of users in organizations, this is a widely deployed approach.
- Extensions of RBAC: These are many: LRBAC [25], CRBAC [20], CA-RBAC [13], OrBAC [1], etc. They have been designed to deal with situations not considered by RBAC such as changeable contexts, mobile computing, pervasive computing, etc.
- The Attribute based access control (ABAC) [25] approach, that generalizes the concept of context by introducing the "attribute concept".
- New access control approaches have emerged in the Cloud to tackle access control requirements such as: HABAC for hierarchical ABAC model for Cloud storage [23], and Task RBAC with a constraints model for Cloud provisioned healthcare systems [18].

As we can see, formal access control models are evolving along with technology usage and user requirements. Indeed, access control models are related to existing security concerns at the time of their conception. Thus, classical access control models cannot easily deal with systems' evolution. We should then move towards generic modeling languages that can be easily adapted to capture this evolution. *Hybrid* models, that combine characteristics from existing models, should be allowed.

Moreover, in the Cloud, users' data are stored in Cloud servers. These data can be accessed by users that are geographically distributed, with different operational constraints. For example, the working hours can differ from a site to another; consequently, the high level policy of companies must be abstract and hide the details which are specific to each site. Such a policy can contain the following rule: "doctors can consult patients' files during working hours". The working hours must be instantiated according to actual values for each site.

In this paper, we propose a new method for the specification of access control in Cloud environments. This method is based on a generic access control meta-model called CatBAC together with a refinement technique. The CatBAC *meta-model* allows Cloud providers to formalize the high level policy of each of their customers by offering each of them an access control *model* specific to this policy. This access control model makes it possible for security administrators in the various sites of the company to produce a *concrete model* taking into account the constraints and specificities of each site. The passage between meta-model, model and concrete model is ensured by the refinement process. The concrete model represents the access control policy in each site and respects the high level policy of the company. Although we will show that our technique is particularly useful for the design of Cloud security services, it is not difficult to see its applicability in other situations where flexible security services are required. The method developed in this paper is an application of the principles introduced in [11].

The rest of the paper is organized as follows. In Section 2, we describe our method for specifying access control in Cloud environnements. In Section 3, we present the generic meta-model CatBAC for hybrid access control policies based on the categorization of basic elements of access control. In Section 4, we describe a refinement technique for building access control models from meta-models. In Section 5, we describe the usage of our method in Cloud environments through a Cloud Scenarios. Section 6 compares the work presented in this paper with the relevant work in the literature. We draw conclusions for this paper in Section 7.

2. OUR METHOD

The CatBAC meta-model is based on the concept of category and makes it possible to formalize hybrid policies which can combine several elements of the classical access control models.

In order to protect the data of a given company, the Cloud provider initially recovers the high level policy of this company which must be abstract. The meta-model will be refined in an abstract model of access control representing this policy. This abstract model will be sent to the various sites of the company allowing the local security administrators to refine it in a concrete model by respecting the low level security policy and the constraints locally required. The concrete model will be used to treat the coming access requests from its corresponding site. Figure 1 shows our method.

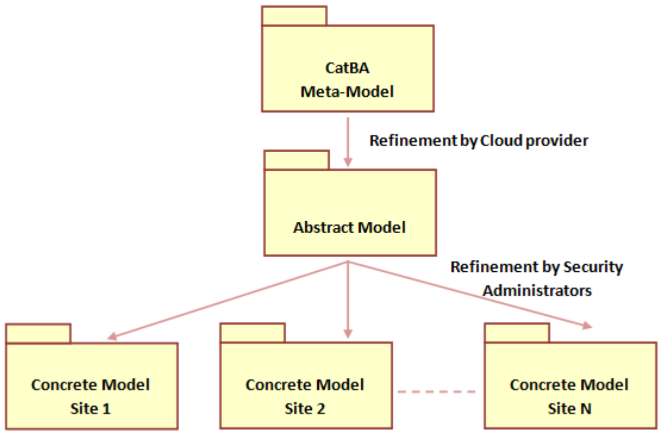

Figure 1: Our Method

An implementation of our CatBAC framework could be offered as a Cloud service. We will demonstrate two advantages for this method: (1) It allows to express hybrid policies, (2) It makes access control more flexible allowing to take into account the local constraints and specificities of each site of a company. The Cloud provider refers to the concrete model of each site to treat incoming access requests.

3. GENERIC META-MODEL CATBAC

In this section, we present our meta-model CatBAC and we show how it can be used to define hybrid access control policies. Beginning with the basic access control concepts, we propose a new approach to represent these concepts in a more generic way.

3.1 Basic access control elements

Basic elements of access control are: subjects (S) which access resources (R) to perform actions (A). An access control rule associates these elements to express permissions or prohibitions. For example: "Alice can read file F1".

By adding contextual conditions, rules and policies can become more expressive. For example: "Alice can read file F1 between 8:00 am and 4:00 pm", or "Doctors can read medical files of all patients in emergency situations". Context (C) is an important element to provide realistic policies for access control. Hence, access control in its most concrete form is a relationship between these four elements.

3.2 CatBAC meta-model

Many access control models and recent meta-models proposed in the literature are based on the notion of "Category". In RBAC [10], *role* can be considered as a unique category for classifying subjects. In MAC [14], there is a *Security level* categorization for both subjects and resources. Recently, Barker [3, 4] has proposed a generalization of the category concept in RBAC by considering a large number of categories such: roles, groups, security levels, localization, etc. Thus, the category concept can be used to build more generic access control models.

However, in previous work [3, 4, 10, 24], the concept of category was applied to specific elements. In this new approach, we generalize the use of the category concept and apply it to all basic elements of access control, as shown in Figure 2. In this way, we have an abstract level that connects categories to express authorizations (permission, prohibition), that we call "abstract authorizations". On the concrete level, "concrete authorizations" which represent the access control decisions in relation to concrete entities (subject, resource, action and context), are based on the assignment of these entities to categories and on the existence of an abstract authorization between these categories.

On this basis, we can define three types of relations for modeling access control:

1. *Abstract authorization*: this relation associates the four categories (subject category (scat), resource category (rcat), action category (acat) and context category (ccat)) to express modalities of access control such as: permission, prohibition, etc.

2. *Assignment relation*: this relation associates concrete elements of access control to their corresponding categories to express low-level policies.

3. *Concrete authorization*: this relation associates concrete elements to express concrete decisions of access control; it is based on the two relations presented above.

Following Barker [3], we can formalize our access control mechanism through first-order logic. We define the relationship between the abstract and concrete levels by the axiom:

$$\text{A-Auth(scat, acat, rcat, ccat)} \wedge \text{SCat(s, scat)} \wedge \\ \text{ACat(a, acat)} \wedge \text{RCat(r, rcat)} \wedge \text{CCat(c, ccat)} \wedge \\ \text{Valid\_Context(s, a, r, c)} \Rightarrow \text{C-Auth(s, a, r)}, \text{ where:}$$

- A-Auth(scat, acat, rcat, ccat): defines an abstract authorization between categories scat, acat, rcat and ccat.

- XCat(x, xcat) for X $\in$ {S,A,R,C} and x $\in$ {s,a,r,c} is a predicate that is true iff x $\in$ xcat.

- Valid_Context(s, a, r, c): is a predicate that tells us about the validity of context c for concrete elements s, a and r. If the context is valid, this predicate returns *true*.

- C-Auth(s, a, r): defines a concrete authorization between concrete elements s, a and r, stating that subject s can or can't perform action a on the resource r.

In the terminology of access control, the notion of authorization is used to indicate either permission which is a "positive authorization", or prohibition which is a "negative authorization".

CatBAC (Figure 2) is a UML-based meta-model for specifying hybrid access control policies. It is based on the abstract level relation. It is composed of four abstract UML classes: SCat (Subjects Categorization), RCat (Resources Categorization), ACat (Actions Categorization) and CCat (Contexts Categorization). These classes are abstract classes that can be refined into more specific classes. These specific classes produce a CatBAC model derived from the CatBAC meta-model, in order to implement the high-level access control requirements.

The classes in the CatBAC meta-model can be instantiated to sets of classes representing each a category concept that can be used to classify concrete elements of access control. In a generic way, we consider that these categories can regroup many types of categories used in access control specifications that can cover novel categorization types. For subject's categorization, many categories are known, such as: roles, groups, security levels, etc. On the other hand, categorization of resources by security levels is one of the main characteristics of the MAC model. We generalize this by proposing other categorizations of resources such as: groups, risk value, etc. In the same way, we define categorizations of actions to classify them with respect to certain criteria, like: groups, sensitivity level, risk value, etc. CCat represents categorization of contextual information such as time, localization, session, situations, etc. Figure 2 represents, in UML notation, the four classes in CatBAC.

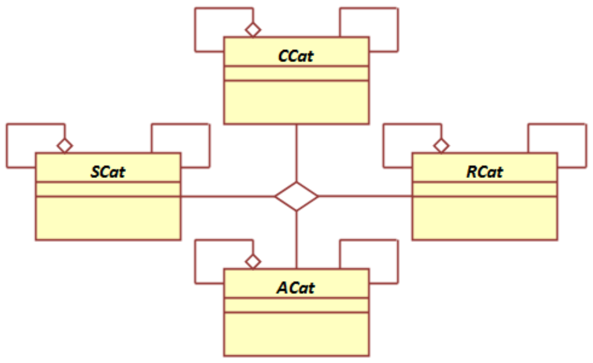

Figure 2: UML representation of CatBAC meta-model

The self-association edge on each of the four classes defines the possibility of allowing categories to be associated with other categories, which allows a high level of hybridization between multiple access control models. For example, the role category can be categorized by groups. Also, we define hierarchical relationships among categories by the aggregation relation. This is shown as a self-association edge with a diamond head. For example, the category for the location context can include a geographical hierarchy among countries and continents such as: "Bavaria is in Germany which is included in Europe".

The CatBAC meta-model has the advantage of being symmetric with comparison to other models proposed in the literature [3, 4, 24]. This symmetry allows a "clean" separation of the abstract level from the concrete level; in addition, the formalization of the model will be simpler.

In Role-based access control (RBAC), when a new subject is assigned to the *system*, it is assigned only the roles the subject will play to benefit from the privileges associated with these roles. We adopt this principle to all basic access control elements (subject, resource, action and context) to further facilitate the administration when there is addition of new elements. For example, when a new resource is added into the system, it is assigned its corresponding category of resource. Therefore, fewer authorization rules will need to be added. Also, offering several categories for a given class of access control elements (subject, resource, action or context) allows combining several access control models, and therefore producing hybrid policies that can meet new security requirements. In section 3.4, we present a simple example of hybrid policy.

3.3 Deriving CatBAC models

In this section, we show how classical, new and hybrid access control models can be refined from the CatBAC meta-model by specializing abstract classes to represent concrete categories. For instance, XCati in Figure 3 represents a real category of basic elements of access control such as: Consultant (as Role), Doctors (as Group), Classified (as Security level), etc.

Classical models of access control can be derived from our meta-model by specializing abstract classes to represent elements of these models. When a classical model doesn't support categorization for a specific set of basic elements of access control, we assume that each element in this set represents a category in itself (XCati = X).

To support hybrid access control models, many categories can be combined from the meta-model.

Therefore, a company may instantiate the meta-model so that it represents its structure and its management of access control. This instantiation can produce an access control model that responds to specific security needs. The generated model represents the access control policy within the company.

For example, the high level policy of an organization may require the following rule to be enforced: "Employees which are consultants and have security level 'Classified' can read File_A". This policy requirement is presented by refinement of the CatBAC meta-model in an abstract and hybrid model (Figure 4). In section 4, we present a technique for refinement that we use with CatBAC.

3.4 Low-level policies

CatBAC allows visualizing low-level policies through UML designed models derived from CatBAC models. Let us suppose that the low level policy contains this rule: "User Alice is a consultant and has security level 'Classified' can read File_A". Then, this low level rule can be visualized using an UML concrte model (Figure 5), that it is a refinement of the CatBAC model presented in Figure 4.

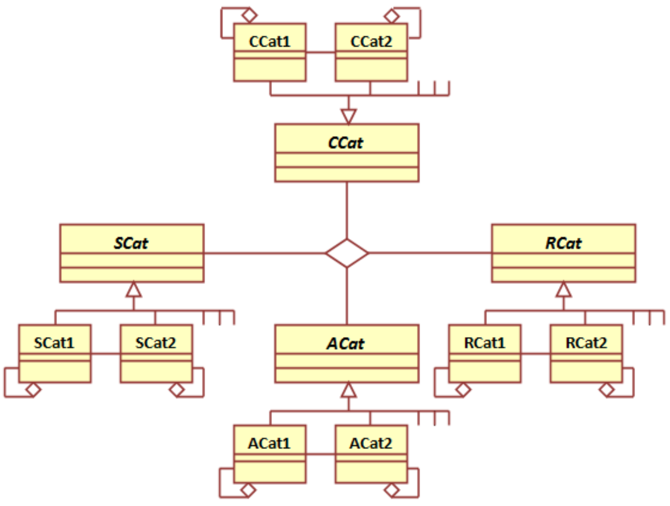

Figure 3: Derivation of hybrid CatBAC models from meta-model

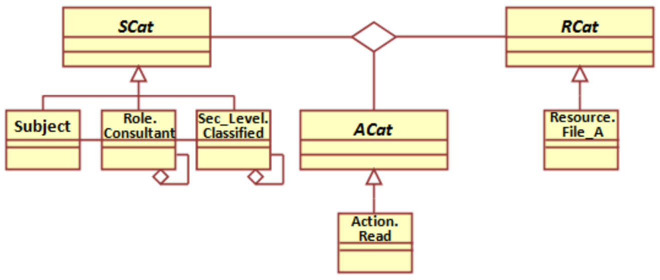

Figure 4: CatBAC model

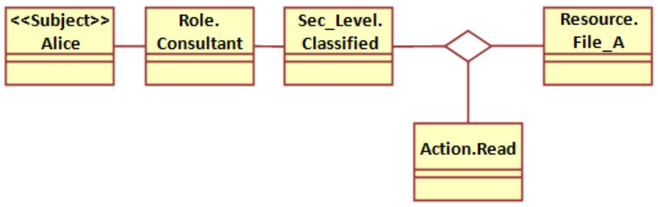

Figure 5: CatBAC concrete model

4. A REFINEMENT TECHNIQUE FOR CATBAC

4.1 Refinement process

As mentioned, the derivation in CatBAC from meta-model to model necessitates a refinement technique. Refinement is the process of transforming abstract or high-level specifications into low-level, concrete ones [17]. Refinement ensures that all elements, namely classes of models are correctly derived from the elements of their meta-model(s). The main objectives of any policy refinement technique are:

1. Determine the resources that are needed to satisfy the access control requirements of the abstract policies,

2. Translate abstract policies into low-level policies,

3. Verify that low-level policies actually meet the requirements specified by their corresponding abstract policies.

In our case, the refinement process involves two steps: high-level refinement and low-level refinement.

4.1.1 High level refinement

The high level refinement produces an access control model satisfying the high level policy of the organization, while respecting the meta-model. It produces the mechanisms used to manage access control in accordance with the organization's policy. We will illustrate this by means of an example. Suppose that the high level security policy of an organization is stated as follows: "Employees who are *consultants* can *consult* documents in the group *Financial_Documents* during *Working Hours*". The process of high level refinement can, then, be decomposed into six steps:

1. Extraction of values of the categories: these are the values of categories used to manage access control in the organization. For example, values such as: *Consultant*, *Financial_Documents*, *Consult* and *Working_Hours* can be extracted from the policy of our example.

2. Define types of categories: for each value, determine the type of categorization used (couples (Type, Value)). For example: *Consultant* is a *Role*, *Financial_Documents* is a *Group* of resources, *Consult* is a *Group* of actions and *Working_Hours* is a *Time*. Then, we can determine couples (Type, Value) of different categorizations used in the organization : (*Role*, *Consultant*), (*Group*, *Financial_Documents*), (*Group*, *Consult*) and (*Time*, *Working_Hours*). Note that the intervention of experts or administrators within the organization is needed to correctly define the type of categorization, because, for example, *Consultant* can be considered as a *Group* of subjects in another organization.

3. Determine concrete entities: for each couple (Type, Value) of categorization, determine the corresponding set of basic entities (Subject, Resource, Action or Context). For example: (*Role*, *Consultant*) is a categorization used for *Subjects* (Employees in our policy), (*Group*, *Financial_Documents*) is a categorization used for *Resources*, (*Group*, *Consult*) is a categorization used for *Actions* and (*Time*, *Working_Hours*) is a categorization used for *Context*.

4. Construct the category set for each basic entity by enumerating concrete categories used for this entity; we use `Type.Value` to express categories in comprehensive way. We call this set *H-Ref*.

   $$\text{H-Ref: Basic Entity} \rightarrow \text{Categories}.$$

   H-Ref(Subject) = {Role.Consultant},
   H-Ref(Resource) = {Group.Financial_Documents},
   H-Ref(Action) = {Group.Consult},
   H-Ref(Context) = {Time.Working_Hours} in our example.

5. Add to each H-Ref(X) set, the basic entity name X.

   $$\text{H-Ref(X)} = \text{H-Ref(X)} \cup \{X\},$$
   $$\text{with: } X \in \{\text{Subject, Action, Resource}\}.$$

   because each set of entities is a special categorization where each element represents a category itself. When for a basic entity X, no categorization is specified, we take the concrete value for this entity specified in the high-level policy, such as a concrete category value (which contains a single element). So we add `X.ConcreteValue` to H-Ref(X).

   $$\text{H-Ref(X)} = \text{H-Ref(X)} \cup \{\text{X.ConcreteValue}\}.$$

   By applying this step to our example, we get:
   H-Ref(Subject) = {Role.Consultant, Subject},
   H-Ref(Resource) = {Group.Financial_Documents, Resource},
   H-Ref(Action) = {Group.Consult, Action},
   H-Ref(Context) = {Time.Working_Hours}.

6. Specialize the super classes XCat of the meta-model into a set of classes corresponding to the elements of sets H-Ref to generate the corresponding CatBAC model. In our example, we specialize, for example, the SCat (Subject's Categorization) class into classes `Role.Consultant` and `Subject` by respecting elements in H-Ref(Subject).

The transformation from Figure 2 into Figure 4 exemplifies this process. In section 5, we present an example of the refinement process for a healthcare scenario with UML representation of meta-model, abstract and concrete models.

4.1.2 Low level refinement

Low level refinement allows to instantiate the classes of the CatBAC model with actual values extracted from the concrete policy. So each concrete policy is represented by a concrete model.

The classes of the abstract model representing concrete category values (Type.Value) or concrete values of basic entities (X.ConcreteValue) will be preserved in this level of refinement. Other classes will be refined by the concrete values expressed in the low-level policy.

4.2 Refinement representation

Refinement captures the essential relationship between specification and implementation. Policy refinement has been studied in many different research contexts such as [9, 2]. We have chosen to use the method of [9], since it proposes a simple way to refine UML based models.

This method proposes an extension of UML to express complex model refinement by means of a well-defined composition of elementary refinements. The extension defines stereotypes expressed in OCL [19]. In this extension, each refinement relates two (or more) elements: the abstract and the refined ones. The stereotype "refinedElement" is the "path" to express what internal elements of a package - which regroups UML elements (classes, objects, etc.) - are refined, and a "mapping" to specify the way the refinement is made. The stereotype "refine" is a refinement relation between packages representing each model.

5. USE OF OUR FRAMEWORK IN CLOUD ENVIRONMENT

5.1 Case Study: Electronic Medical Record

A Cloud provider offers Software as a Service (SaaS) for a healthcare organization "New_Hospital" which contains several distant sites. This software allows doctors who are radiologists to view their patients' electronic medical records (EMR) by using web browsers. For example, when doctors open their browsers, they log in as radiologists, and then the information on their patients is displayed: names, diagnosis, lab results, imagery and other related information. This information is stored in Cloud servers provided by the Cloud provider. Hence, the Cloud provider needs to offer access control solutions to protect patient EMR data and ensures that only doctors that were able to authenticate themselves as radiologists for specific patients can access the EMR of these patients. Moreover, the Cloud provider must take into account the various local security requirements imposed in each site of this organization.

More precisely, in order to protect patients' EMR data from unauthorized access, the Cloud provider needs to implement the following abstract high level security policy specified by organization "New_Hospital": *"doctors who are radiologists can read their own patients' EMR data, and this data is classified as private; constraints on location or time can be applied"*. We can conclude that the following access control concepts must be taken into consideration: the job function of the user in the organization, known as *Role*, the *Security Level* concept defined by a MAC model to specify data classification, and *Context* type information like *Location* and *Time* to specify, if necessary, users' physical location and time of access. Extraction of access control requirements will be achieved by the high-level refinement process.

On the other hand, security administrators who are responsible for managing the security in their sites are considered to be Cloud users. They are responsible for specifying the low-level access control policies and enforce them in Cloud access control systems offered by Cloud providers. For example, a security administrator can specify the following access control policies that respect the abstract security policy of his/her organization:

Policy1 : "Radiologist Alice can read the medical record EMR1 of her patients classified as private only between 8am and 5pm",

Policy2 : "Radiologist Bob can read the medical record EMR2 of his patients only when he is in the hospital".

In the following section, we will show how to apply this scenario in our formal framework.

5.2 Application of the refinement technique

The abstract security policy can be specified by the Cloud provider as the global policy for organization "New_Hospital" for high-level requirements. This abstract security policy is represented in our framework CatBAC as a model that is refined from our meta-model using the UML extension [9] as seen in Figure 6. Extraction steps of access control requirements by the high-level refinement (section 4.1.1) are summarized in the following table:

Table 1: Categories extraction

Values	*Categories*	*Basic Entities*
Radiologist	Role	Subject
Private	Sec_Level	Resource
EMR	Group	Resource
Read	-	Action
Time	Time	Context
Location	Location	Context

Note that no categorization is used for actions. In the organization policy, the concrete action *read* is specified and it is taken in consideration in low level policies, see below. For Context entity, two categories are specified: Time and Location, but no value for these categories was given. Real values are left to the administrators to express local constraints applied to various users' access.

We define sets H-Ref for all basic entities:

- H-Ref(Subject) = {Role.Radiologist},
- H-Ref(Resource) = {Sec_Level.Private, Group.EMR},
- H-Ref(Action) = {}, (no categorization for Actions)
- H-Ref(Context) = {Time, Location}.

By adding the basic entities sets, we obtain the sets of refinement classes as follows:

- H-Ref(Subject) = {Role.Radiologist, Subject},
- H-Ref(Resource) = {Sec_Level.Private, Group.EMR, Resource},
- H-Ref(Action) = {Action.Read},
- H-Ref(Context) = {Time, Location}.

5.3 Representation in CatBAC

Notice that we have used the stereotype "refine" that exists in UML to define that the package containing the model is refined from the package that contains the meta-model. Furthermore, the internal element such as the class of the package of the model is correctly refined from the corresponding internal element of the package of the meta-model using stereotype "refinedElement, which_element". The part 'which_element' indicates only which class this internal element is refined from.

For instance, classes `Subject` and `Role.Radiologist` are refined from class `SCat`, class `Sec_Level.Private`, `Group.EMR` and `Resource` are refined from `RCat`, class `Action.Read` is refined from class `ACat` and classes `Time` and `Location` are refined from class `CCat`. This allows us to represent a hybrid access control that is a combination of different access control models that define the security concepts role, context, and security level. Note that this involves security concepts coming from RBAC, OrBAC and MAC, among others.

Now security administrators can specify their access control polices or low-level policies as refinements of the above model. Each low-level policy is represented using a design model that is correctly refined from the above CatBAC

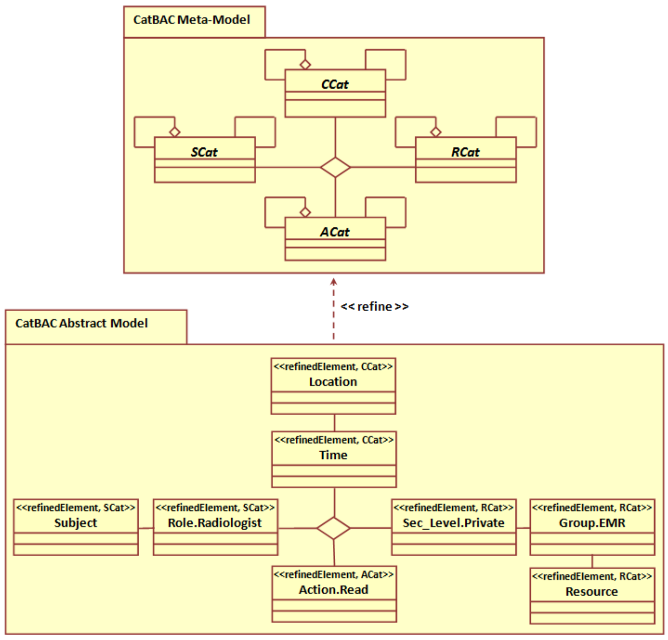
CatBAC Meta-Model
CCat
SCat
RCat
ACat
<< refine >>
CatBAC Abstract Model
<<refinedElement, CCat>> Location
<<refinedElement, CCat>> Time
<<refinedElement, SCat>> Subject
<<refinedElement, SCat>> Role.Radiologist
<<refinedElement, RCat>> Sec_Level.Private
<<refinedElement, RCat>> Group.EMR
<<refinedElement, ACat>> Action.Read
<<refinedElement, RCat>> Resource

Figure 6: Refinement of the model from meta-model

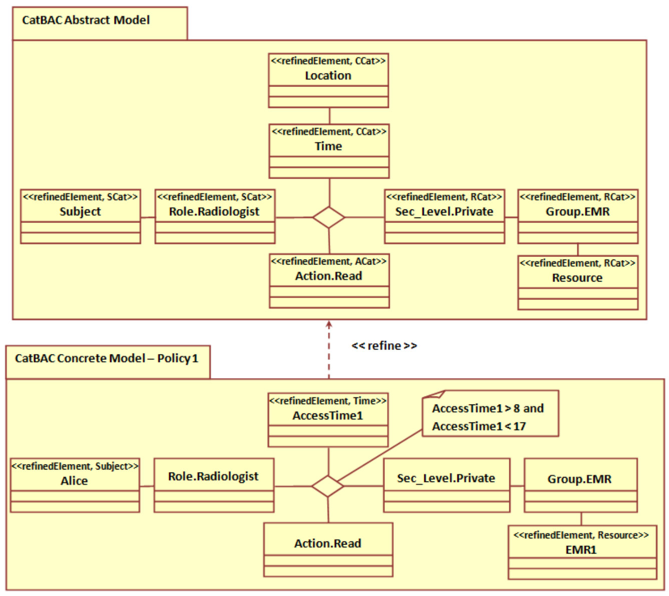
CatBAC Abstract Model
<<refinedElement, CCat>> Location
<<refinedElement, CCat>> Time
<<refinedElement, SCat>> Subject
<<refinedElement, SCat>> Role.Radiologist
<<refinedElement, RCat>> Sec_Level.Private
<<refinedElement, RCat>> Group.EMR
<<refinedElement, ACat>> Action.Read
<<refinedElement, RCat>> Resource
<< refine >>
CatBAC Concrete Model – Policy1
<<refinedElement, Time>> AccessTime1
AccessTime1 > 8 and AccessTime1 < 17
<<refinedElement, Subject>> Alice
Role.Radiologist
Sec_Level.Private
Group.EMR
Action.Read
<<refinedElement, Resource>> EMR1

Figure 7: Refinement of low-level policies in CatBAC – Policy 1

model. For example, Policy1 is represented in the CatBAC design model as seen in Figure 7. The class `Alice` represents the name of the subject which is refined from class `Subject`. The context information `AccessTime1` is refined from class `Time`. By using OCL, we can represent the context constraint on the time when Alice can access medical records of her patients. Furthermore, `EMR1` is refined from class `Resource`.

6. RELATED WORK

Some Cloud-oriented access control methods have been proposed in the literature, among others HABAC [23] and Task RBAC [18]. Most such methods are based on extensions of one of the existing models, for example HABAC is based on XACML and Task RBAC is based on RBAC. However, we cannot assume that access control models such as these will remain stable over time. The evolution of the technology will lead to the introduction of new formal access control models. As proposed by Barker [3], it seems appropriate to define a modeling language that is capable to represent these different, yet unknown, access control models.

Recent research has also resulted in the development of UML-based modeling languages that incorporate different security requirements coming from traditional access control models into system design models [5, 21, 22]. These developments have motivated our approach that provides support for modeling the basic concepts of the DAC, MAC, RBAC, ABAC models, as well as combinations of them, based on the category notion. Our meta-model is a generalization of the work of Barker [3, 4] since it proposes a categorization of all the basic elements of access control while adding the notion of context, which does not appear in the formalism of Barker. In addition, our meta-model is open to evolution and can support new models of access control.

Work that is close to ours on the conceptual level is that of OrBAC [1], together with its extension in [6]. OrBAC proposes an abstract level between role, activity and view. For us, these are specialized categories in CatBAC. Consequently, our meta-model can represent OrBAC policies. OrBAC includes the notion of context; we propose that context be expressed as a category to obtain a symmetric, simple and intuitive meta-model.

7. CONCLUSION

We have presented a method for the design of security Cloud services. This method allows specifying access control models to secure access to data stored in the Cloud. Cloud providers must offer to their users security mechanisms that protect them from inappropriate access to their sensitive data stored on Cloud servers. Given the many possible applications of Cloud servers, these security mechanisms must be capable of being configured in many ways.

Our method is based on CatBAC, a generic meta-model of access control, with two stages of refinement. In the first stage, the meta-model is refined into an abstract model according to the high-level policy of the organization, this stage is completed by the Cloud provider. The second stage allows to refine the generated abstract model in several concrete models by network administrators at the various sites of the organization, and this by respecting the local constraints and specificities of each site.

The method illustrated in this paper gives Cloud providers an access control security solution that can be seen as a Cloud service for both providers and users. It allows users to define their own low-level policies in a way that these policies can be refined correctly from the abstract policy defined by their Cloud provider.

An important point on which we cannot dwell in this paper is how to verify that the properties of each model are preserved in the combination with other models. This will be the subject of future publications. In future work, we will also explore the application of our method in the context of multi Cloud environments, where applications and resources are deployed across different Clouds and in which Cloud providers may collaborate. This will open the door for the study of problems of trust and interoperability between Cloud providers. Other work will be to develop automatic refinement techniques by using a formatting language for high-level policies.

8. ACKNOWLEDGMENTS

This work was supported in part by grants of the Natural Sciences and Engineering Research Council of Canada. We acknowledge the collaboration of Marwan Cheaito in earlier phases of this work.

9. REFERENCES

[1] A. Abou-El-Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131, 2003.

[2] B. Aziz, A. Arenas, and M. Wilson. *Model-Based Refinement of Security Policies in Collaborative Virtual Organisations*, volume 6542 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 2011.

[3] S. Barker. The next 700 access control models or a unifying meta-model? In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 187–196, 1542238, 2009. ACM.

[4] S. Barker. *Logical Approaches to Authorization Policies*, volume 7360 of *Lecture Notes in Computer Science*, pages 349–373. Springer Berlin / Heidelberg, 2012.

[5] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91, 2006.

[6] Y. Bouzida, L. Logrippo, and S. Mankovski. Concrete- and abstract-based access control. *Int. J. Inf. Secur.*, 10(4):223–238, 2011.

[7] R. Buyya, Y. Chee Shin, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 5–13, 2008.

[8] M. Carvalho. Secaas-security as a service. *ISSA Journal*, pages 20–24, 2011.

[9] N. Correa and R. Giandini. A uml extension to specify model refinements. In *CLEI 2006*, 2006.

[10] D. Ferraiolo and D. Kuhn. Role-based access control. In *15th Nat'l Computer Security Conf.*, pages 554–563, 1992.

[11] S. Khamadja, K. Adi, and L. Logrippo. An access control framework for hybrid policies. In *Security of Information and Networks, 2013. SIN '13. The 6th International Conference on*, 2013.

[12] R. L. Krutz and R. D. Vines. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing.* John Wiley & Sons, 2010.

[13] D. Kulkarni and A. Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 113–122, 1377854, 2008. ACM.

[14] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. The inevitability of failure: The flawed assumption of security in modern computing environments. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314, 1998.

[15] Y.-G. Min, H.-J. Shin, and Y.-H. Bang. Cloud computing security issues and access control solutions. *SERSC: Journal of Security Engineering*, 9:135–142, 2012.

[16] P. Mirchandani. Security-as-a-service - the next growth area for cloud computing?, Oct 26 2009.

[17] J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed systems management. *IEEE J.Sel. A. Commun.*, 11(9):1404–1414, 2006.

[18] H. A. J. Narayanan and M. H. Gunes. Ensuring access control in cloud provisioned healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 247–251, 2011.

[19] OMG. Object management group. object constraint language, version 2.2, 2010. omg document number: formal/2010-02-01.

[20] S.-H. Park, Y.-J. Han, and T.-M. Chung. *Context-Role Based Access Control for Context-Aware Application*, volume 4208 of *Lecture Notes in Computer Science*, pages 572–580. Springer Berlin / Heidelberg, 2006.

[21] J. A. Pavlich-Mariscal, S. A. Demurjian, and L. D. Michel. A framework of composable access control features: Preserving separation of access control concerns from models to code. *Computers & Security*, 29(3):350–379, 2010.

[22] I. Ray, N. Li, D.-K. Kim, and R. France. *Using Parameterized UML to Specify and Compose Access Control Models*, volume 140 of *IFIP International Federation for Information Processing*, pages 49–65. Springer Boston, 2004.

[23] L. Shi-Xin, L. Feng-Mei, and R. Chuan-Lun. A hierarchy attribute-based access control model for cloud storage. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, volume 3, pages 1146–1150, 2011.

[24] N. Slimani, H. Khambhammettu, K. Adi, and L. Logrippo. Uacml: Unified access control modeling language. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1–8, 2011.

[25] H. Zhang, Y. He, and Z. Shi. *Spatial Context in Role-Based Access Control*, volume 4296 of *Lecture Notes in Computer Science*, chapter 15, pages 166–178. Springer Berlin Heidelberg, 2006.