DESIGN ARCHITECTURE FOR UCON MODEL IN CLOUD ENVIRONMENT

NAZIYA PARVEEN 1*, MOHD ANWAI ALI 2*, MNA BAIG 3*
1. II-M.Tech (CS), A.H.C.E.T,HYD.
2. ASSOCIATE PROFESSOR, A.H.C.E.T, HYD.
3. ASSOCIATE PROFESSOR, A.H.C.E.T, HYD.

Abstract: Today modern computing systems leverage distributed models such as cloud, grid, etc. One of the obstacles of wide spreading these distributed computing models is security challenges which includes access control problem. These computing models because of providing features like on-demand self-service, ubiquitous network access, rapid elasticity and scalability, having dynamic infrastructure and offering measured service, need a powerful and continuous control over access and usage session. Usage control (UCON) model is emerged to cover some drawbacks of traditional access control models with features like attribute mutability and continuity of control. Several recent works have been done to apply UCON for distributed computing environments, but none of them could cover all aspects of the model. In this paper we propose an architecture for applying UCON model in cloud environments. Moreover we present a new architecture for obligation handling. We also introduce a new approach to handle attribute mutability. For implementation we have extended XACML syntax and semantics as policy language and leveraged Sun’s OASIS XACML implementation.

1. INTRODUCTION

Cloud computing is a new computing model which delivers computing resources as a service to users. According to NIST definition of cloud computing [1]:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Protecting cloud resources from unauthorized access and policy violations is a primary need for using clouds, so leveraging a powerful access control model is essential.

In general, access control means applying a system to monitor all requests and evaluating them against existing policies and making a decision whether to permit access or to deny it.

As discussed in [2, 3], traditional access control models such as MAC, DAC, RBAC, as well as newer models such as TM and DRM, cannot overcome new challenges which distributed computing environments like clouds may encounter.

UCON proposed by Park and Sandhu [4, 5], is a novel usage control model that augments traditional access control models and also TM and DRM by features such as attribute mutability and continuous control.

Gouglidis et al. [6] present the cloud access control requirements by means of a four-layer conceptual categorization. They argue that for all requirements except those related to management layer of categorization, UCON model is the best solution. However up today a comprehensive and precise implementation of UCON has not been presented and the model has remained at conceptual level. In this paper, we advocate the adoption of UCON model in cloud computing environments and propose a full reference architecture for any implementation of UCON.

The remainder of this paper is organized as follows. Section II concentrates on the UCON model and its components, section III describes the background and the related works, section IV introduces the proposed UCON architecture, section V discusses proposed obligations handling model, section VI explains our implementation of the proposed architecture which is based on XACML language. Finally we conclude the paper and outline the ongoing and future works in section VII.

II. UCON MODEL

The main concentration of usage control is controlling the user usage session after granting access permission to him. In other words in comparison with access control approaches which target pre-usage controls, usage control offers new features to handle dynamic aspects of new computing environments during usage. Two novel features of UCON include: 1) mutability of attributes, which means UCON model can handle attributes updates which take place as a side effect of usage, 2) continuity of control, which means access decision will be evaluated during usage.
UCON model is an attribute based model, it means access permissions on resources are based on predicates on subject, object or environment attributes and are defined in the form of authorization, condition and obligation policies.

The main components constructing UCON model are shown in Fig. 1.

Subjects are entities which request to access some objects and can do some actions on them according to their rights. Subjects and objects will be defined by their attributes.

UCON model decision making is based on the evaluation result of authorization, condition and obligation policies.

Authorization policies define some constraints on subjects and objects attributes. In UCON model, authorization policies can be evaluated before access or during usage. These kinds of policies are the only decision factor in traditional access control models.

Conditions are constraints on environment attributes such as system time, CPU-ID, running program platform, etc. Same as authorization policies, condition policies can be evaluated before access or during usage. Obligations are mandatory actions that should be fulfilled by system or obligation subjects. According to who should do the obligation action and whether the obligation is observable or not, many suggested categorizations are proposed, such as [7, 8].

According to evaluating authorization, obligation or condition policies (three modes), before or during usage (two modes) and do attribute update actions before, during or after usage or does no attribute update (4 modes), 24 core UCON models will be defined [9].

III. RELATED WORKS

In this section, we have outlined prior publications which are most related to our research and we state their challenges and give a comparison with our approach.

Katt et al. [7] proposed a classification on obligations and then introduced an extension on UCON state transition diagram. They presented a general architecture for UCON and a model for obligation enforcement. In comparison with our work, they integrated the policy enforcement point module in the application, so it was not in a general manner that can be referenced from outside and can act independent of application. Moreover in their proposed model, the details of inner architecture of the obligation module has not been elaborated.

Furthermore they did not suggest extensions on XACML to cover UCON requirements.

Colombo et al. [10] extended the XACML and its proposed OASIS architecture to support UCON’s new features. They described possible methods to do periodic checks for handling attribute mutability but did not choose suitable one. Additionally, they did not suggest obligation enforcement architecture and fulfillment check solution.

Zhang et al. in [11] proposed a usage control enforcement architecture in the basis of trust. Similar to our approach, the main goal of their work and proposed model is to have control over data after being released to users. Their work relies on trusted subsystem in which all enforcement components are in client side. On the other hand our main focus is covering all UCON features for cloud environments in which the control service resides in a remote server. So research efforts targeting different problems.

Zhang et al. in [12] presented a UCON based security framework for collaborative computing systems. Similar to our work they extended XACML for implementation but they just considered pre-obligations and have not proposed an architecture for obligation enforcement.

Gama et al. [13] introduced a platform for obligation enforcement which resides below runtime application and acts independent of that (Heimdall). They do not consider the whole usage session and the relation of obligation engine with other components.

Danwein et al. [14] proposed an enforcement model for UCON that its main concentration is enforcing privacy policies by means of Negotiation Module. They do not consider the UCON components in detail.

Ali et al. [15] argue that definition and enforcement of obligation policies is one of the key aspects of privacy protection. The contributions of their paper are mainly related to proposing a formal language to express user obligations. Through their work, they pointed general requirements of an obligation language and its enforcement platform, and finally, they proposed an architecture to enforce obligations.
Demchenko et al. [16] independent of UCON model, proposed a reference model for obligation handling (OHRM) by means of extending current XACML policy enforcement model. But the introduced model can be used just for system obligations and other kinds of obligations, such as subject obligations, have not been considered. Additionally in comparison to our work they just cover obligation component of UCON in a limited manner.

IV. ENFORCEMENT ARCHITECTURE

As mentioned earlier, the major challenge in using UCON model is lack of a comprehensive architecture to cover whole components of this model and satisfy its novel features. Our proposed architecture is shown in Fig. 2.

In the following paragraphs we aim to discuss the responsibility of each inner component of the architecture.

Figure 2. A UCON based architecture for cloud environments

A. Enforcement Point

General task of this component is issuing decision requests to Decision Point module and also customizing and enforcing decisions. So any requests for accessing cloud service, cross through this component. The other responsibilities of this component have been described along with the other components explanation.

B. Decision Point

General task of this component is evaluating policies and delivering authorization decision. In proposed architecture, this component is the main part of reference monitor and evaluates every incoming request from Enforcement Point by means of existing policies in Policy Manager’s policy database, which includes authorization, condition and obligation policies. In the policy evaluation process, Decision Point communicates with Attribute Manager for subject, object and environment attributes acquisition. Finally Decision Point sends the results, containing decisions made and obligations, to Enforcement Point and obligation Handler respectively.

C. Context Handler

This component same the Context Handler in the OASIS proposed architecture for implementing XACML [17], converts concepts of a context to another. In other words it is used to provide a proper understanding of messages for involved components. These messages contain communications between Enforcement Point with Decision Point, Decision Point with Policy Manager and Decision Point with Attribute Manager.

D. Policy Manager

The responsibility of Policy Manager module is administering authorization, obligation and condition policies and storing them in the policy database. Moreover announcing attribute types to Attribute Manager will be done by this module. Attribute type specifies the way the attribute mutability should take place, we will discuss it in detail in the next paragraph.

E. Attribute Manager

Receiving updated information about subject, object and environment attributes from improvised sensors resides in these elements’ sides and storing them in an attribute database is the responsibility of this component. One of the significant challenges for enforcing UCON model, is implementation of attribute mutability trait and continuous control over user usage session. Therefore it is required that Attribute Manager provides updated attributes for the other parts need them.

To provide optimal functionality in the ground of both security and computational overhead, our proposed model performs continuous control according to the sensitivity degree of desired attribute.

For sensitive attributes, Attribute Manager must inform Decision Point when the attribute values would change, this makes Decision Point evaluate the policies again. However for attributes with lower sensitivity degree, Decision Point itself requests Attribute Manager to be informed of updated attribute values in a periodically manner, and if changes has occurred, reevaluation takes place. The gap between each check can be dependent on many factors such as user type, closeness of attribute values to the boundary values and etc.

F. Service

This component shows the service which is requested by user, and by means of that, the user gains access to the protected resources. The user request to access the service would be analyzed by reference monitor and therefore makes decision whether give an access privilege to the user or not, and if so what controls and actions should be done during execution.
G. Event handler

Task of Event handler is to receive any external event from user platform or program execution environment and analyzing and then referring it to the appropriate component. These events include receiving access request to the service, performing an action which causes a change in the state of subject obligations, events which cause policy reevaluation, the end of usage event, network rupture and any other events.

H. Executer

The responsibility of this component is Executing system obligations. To do this, executor receives analyzed system obligations from obligation handler and executes them by planning a scheduler.

I. Obligation Handler

This component receives obligations from Decision Point and processes them, then issue appropriate commands to the related modules. Following section introduces this component in deep details.

V. OBLIGATION HANDLING MODEL

So far, immense architecture of the UCON enforcement model presented and the functionality of each architecture component is explained separately. One of the most important components in the proposed architecture is the obligation handler which we aim to introduce its inner structure in this section. Fig. 3 shows inner architecture of obligation handler.

A. Obligation analyzer and convertor

After analyzing policies by the Decision Point, extracted obligations will be sent to obligation analyzer and convertor module. This module analyzes obligations and converts them to a set of ordered tasks. It does this conversion by means of a knowledge base of obligations connected to it. Also storing the converted obligations in the pending obligations database is one of the module’s responsibilities.

In this database, each record corresponds to an obligation. Each obligation record will be defined by obligations field descriptors. One of these fields is the deadline specified for the obligation fulfillment. For obligations which have no deadline, this field will be set to zero.

After analyzing the received obligations from the Decision Point and storing them into pending obligations database, this module sends a description of subject obligations to the user and also system obligations to the system. This will help the system to plan its scheduler for executing system obligations by means of appropriate related plug-ins.

Before Enforcement Point grants a privilege to the user, it must be awaited to percept results from Obligation Handler if there is an obligation in the output decision made by the Decision Point. After analyzing obligations, if there were no pending obligations with zero deadline, the release command will be sent to the user.

B. Event Analyzer

Event Analyzer checks the incoming events against pending obligations database. It means that for each receiving events, Event Analyzer checks whether there is an obligation in the pending obligations database to be fulfilled because of this event or not. If occurrence of one event implies that one or more obligations will be fulfilled, the Event Analyzer deletes the obligation from the pending obligations database. Moreover if the deadline value of the fulfilled obligation has set to zero, it means that user usage session is waiting, so not only event analyzer deletes the obligation from pending obligations database but also sends release command to the Enforcement Point component.

C. Fulfillment Checker

Fulfillment checker module can communicate with the pending obligations database and also the attributes database. It checks pending obligation’s conditions against the updated attribute values in attributes database and if each one of these obligations has fulfilled, it informs the related parts. If one of the mentioned obligations deadline is set to zero, it means that the user usage session is waiting, so the Fulfillment Checker informs the Enforcement Point and then deletes the obligation from database, however if the obligation’s deadline value is greater than zero, after deleting it from the database, the process would finish. In addition to fulfillment check of each
obligation, this component is responsible for checking whether the deadline of any obligation is expired or not, and if it is so notify the Enforcement Point to block the user usage session and does compensatory actions against him.

VI. IMPLEMENTATION

We selected XACML [17] as our policy language, which is a XML-based fine grained policy language and is widely used in today access control systems. However XACML does not support new features of UCON, such as ongoing policy evaluation and ongoing or subject obligations. So we extended the XACML syntax and semantics.

These extensions includes following points:

• Type attribute for authorization policies to specify whether it does pre checks or ongoing ones. It accepts ‘on’ and ‘pre’ values.
• ReevaluationType attribute for authorization policies, in the case that the policy Type is ‘on’, to specify the type of activating reevaluation process. As we stated in section IV, for sensitive attributes, the Attribute Manager must inform the Decision Point whenever the attribute values change, but for attributes with lower sensitivity degree, the Decision Point itself requests the Attribute Manager for updated attributes in a periodically manner. In the latter case we added a ReevaluationTime attribute too.
• Type attribute for condition policies to specify whether it does pre checks or ongoing ones. It accepts ‘on’ and ‘pre’ values.
• ReevaluationTime attribute for condition policies to specify the time of policy reevaluation.
• Type attribute for obligation policies to specify whether it is system or subject obligation. It accepts ‘sys’ and ‘subj’ values.
• FulfillmentTime attribute which specifies the deadline for obligation fulfillment.

In order to implement the proposed architecture and its constructing components we leveraged Sun’s XACML implementation [18] and extended its modules to support our proposed extension on XACML.

VII. CONCLUSION AND FUTURE WORKS

In this paper we proposed a reference architecture for applying UCON model for cloud computing environments. The proposed architecture covers all UCON components and its novel features. Also we introduced a model for obligation enforcement, which supports all obligation types such as system or subject and pre or ongoing obligations. The proposed architecture covers all real world usage control scenarios.

We presented a new classification on attribute types constructing policies, which causes attribute updates propagation and policy reevaluation take place according to attribute sensitivity. We extended XACML policy language syntax and semantics and corresponding to that, we modified sun’s XACML implementation. We plan to publish the details of XACML and Sun’s XACML implementation extensions in future.

As part of future works we plan to integrate our model with trusted computing concepts and extending our model to be used in client side in a trustworthy manner.

REFERENCES