An Approach and Its Implementation for Cloud Computing Security

Yun Bai¹, Khaled M. Khan² and Chunsheng Yang³
¹School of Computing, Engineering and Mathematics
University of Western Sydney, Australia
Email: ybai@scem.uws.edu.au
²Department of Computer Science and Engineering, Qatar University, Qatar
Email: k.khan@qu.edu.qa
³National Research Council Canada, Ottawa, ON, Canada
Email: Chunsheng.Yang@nrc-cnrc.gc.ca

Abstract—With the increasing popularity of the datacenter provided by Cloud computing, the datacenter security is becoming an important issue. Also there are other issues to be concerned about such as its speed and standard, but data security is the biggest one. When an organization uses a remote datacenter provided by Cloud, it needs to maintain the confidentiality of the outsourced data and ensure only authorized user or client is allowed to access its data resource. In this paper, we investigate the security issue related to datacenter of cloud computing and propose a security approach for it. We use a formal logical method to specify the data and employ intelligent agents to enforce appropriate security rules on it. We outline the authentication mechanism and present a detailed authorization or access control approach for it. The implementation of the proposed approach will be discussed and investigated. The authentication and the authorization mechanisms work together to prevent any unauthorized attempt to data stored in the datacenter.

Keywords: Access Control, Knowledge Representation, Knowledge-Based Systems, Cloud Computing security, Formal Specification

1. Introduction

Cloud computing is a structure that provides efficient and cost-effective service, allows an organization to access applications, services or data that resides in a remote site. In this way, the organization can save cost in terms of infrastructure management, software development and the datacenter maintenance. Some major organizations such as Amazon, Google, Microsoft now offer cloud services to the public. Amazon offers virtual machines and extra CPU cycles as well as storage service. Google offers database, online documents, and other online software. Microsoft provides the organizations with the Window applications, datacenters, database services [20].

However, the introduction of the cloud computing also brings major security concern about the organization’s data hosted in a non-local datacenter. Generally, the success of cloud computing depends on three key issues: data security, fast Internet access and standardization [17]. Among the three issues, the biggest concern is data security. When an organization uses a remote datacenter provided by Cloud, it needs to maintain the confidentiality of the outsourced data and ensure only authorized user or client can access its data resource.

This paper is to address the security issue of the data hosted in the datacenter. We propose an approach to protect the datacenter security by using an authentication and authorization mechanisms. Authentication is a mechanism to identify users legitimacy and to ensure that only the legitimate users are allowed to access the data stored in the datacenter. Authorization or access control is a mechanism to control legitimate users’ access on the data items, it ensures that the users are only allowed to access the data and to perform operations on the data according to the security policy and rules. The implementation of the approach will be investigated and discussed.

Many works such as [4], [25],[24] etc. studied authorizations or access control extensively and a variety of authorization specification approaches such as access matrix [6], [8], role-based access control [5], access control in database systems [3], authorization delegation [14], multiple access control policies [13], procedural and logical specifications [2] have been investigated. Since logic based approaches provide a powerful expressiveness [9] [16] as well as flexibility for capturing a variety of system security requirements, increasing work has been focusing on this aspect. However, how these approaches can apply to cloud computing environment is an area yet to be explored.

There are certain research works have been done in Cloud computing security. [12] discussed Cloud computing
security requirements from a systematic point of view. It attempted to provide a roadmap for researchers on the topic of cloud computing security requirements and solutions. Several criteria and factors were discussed and investigated in achieving cloud computing security. [10] presented a design for a file system to secure file system storage service for Web 2.0 application. [15] proposed approaches to provide catch based security and performance isolation for cloud computing environment. In [22], a hierarchical attribute-based encryption for fine-grained access control in cloud storage services was proposed. This approach provides a method to help enterprise to efficiently share confidential data on cloud server. [21] also discussed a hierarchical attribute-based solution for access control in cloud computing. Even there are many cloud computing security research has been carried out, as far as we know, logic-based formal approaches in this area has not been extensively studied. This paper discusses cloud security issue from formal logic point of view.

In this paper, we investigate cloud computing security by using formal approaches. In section 2, we study the features of the cloud computing and propose a structure for secure datacenter with authentication and access control functions provided, the authentication function is also outlined; in section 3, we present a detailed authorization mechanism by using a formal logical specification, the function and evaluation of authorization request are discussed; in section 4 we discuss the implementation of the authorization function; conclusion and some future work are outlined in section 5.

2. Securing Datacenter Structure

In a cloud computing structure, the organizations using the cloud services are normally referred to as clients. These clients can be located geographically differently. Cloud computing provides various services to the clients [11] such as Software as a service (SaaS), Platform as a service (PaaS). Datacenter is an important service provided by cloud computing. It is a collection of data and servers which the clients subscribe. As an organization using datacenter service, its data and application are located on servers geographically different from its local site. As discussed previously, when a client’s data is housed in a datacenter, it saves the cost of maintaining it. When client from different location needs to access its data, the data seems to be just located locally.

However, since the data is located in a non local site, ensuring the security of the data is not as simple as if housed locally [17]. It is not feasible to enforce the same local security measurement. To ensure safe access to the datacenter, it needs a coordinated security measurement between the datacenters and the clients accessing the datacenter.

For this purpose, we propose an approach to ensure the security of a system such as a datacenter hosted by cloud computing: authentication and authorization. Authentication is used to authenticate the identities of the users, it controls who can access the datacenter. On the other hand, authorization is used to control that the legitimate user only performs legitimate operations on the data once it has been successfully authenticated. The two mechanisms work together to effectively provide the datacenter with secure accesses.

We use \( C_1, C_2, \ldots C_n \) to represent the client servers, each of them acts on behalf of a group of users. The users access the datacenter through their respective client servers. We also use \( AoS \) to represent the authentication server and use \( AoS' \) to act as the authorization server. All users need to register with the \( AeS \) server, their information such as password, identification number, IP address, etc. are stored in a database which is managed by \( AeS \) server. This ensures that only the registered, legitimate users are allowed to access the datacenter. All users access right to certain data is also stored in a database which is managed by \( AoS \) server. This controls that the authenticated users only perform legitimate operations on the allowed data of the datacenter.

Here is the process when a user needs to access certain data of the datacenter. The user requests to its client server, the client server passes the information to \( AeS \). \( AeS \) then checks the user’s credential and request with its database if they match or not. If matches, the authentication is successful, it then passes on to the \( AoS \). \( AoS \) checks the user requested operation and data with its database if they match or not. If the user request is within its specified right, the request is granted, otherwise, it is denied.

We will employ a kind of Kerberos [19] system to carry out authentication function. we will only illustrate the authentication process for one client server \( CS \). All other client servers follow the same procedure for their user authentication.

To protect the messages between \( AeS \) and \( AoS \) servers, we assign both servers a secret key for encrypting and decrypting the messages between them. When a user \( U_1 \) requests to access the data of the datacenter, it needs to be authenticated by \( AeS \) server then subsequently authorized by \( AoS \) server.

\( U_1 \) first sends its request to \( CS \), the \( CS \) carries out a basic checkup, then sends the request to \( AeS \) server, \( AeS \) checks the information \( CS \) supplied about \( U_1 \) against its database information about \( U_1 \) if everything matches correctly. If matches, the authentication is successful, then \( U_1 \) is a legitimate user and is issued a ticket encrypted by the secret key shared by \( AeS \) and \( AoS \) servers. \( U_1 \) does not possess the secret key, hence cannot alter the ticket. It can only pass it to \( AoS \) server for requesting access to the datacenter.

To ensure that each data is only accessed by legitimate users and is manipulated by legitimate operations, \( AoS \) server manages the database about the users entitled access
rights to each data of the datacenter. When the AoS server receives the ticket from $U_1$, it decrypts the ticket by its secret key shared with the AoS server. Then compares the user’s request to its database record about what operations the user can perform on which data. If the request is within the specification of $U_1$’s entitled rights, then $U_1$ can access the datacenter as it requested. Otherwise the request is denied.

3. A Formal Method for AoS

Now, we present the detailed specification, function and evaluation of the AoS server.

In our approach, each client server provides data access service for a group of users. These users access the datacenter generally located at a remote site. We assume that the Internet on which the system relies is safe and sound. For the authorization mechanism, assume each agent manages one client server, a master agent takes overall control of all the agents. We concentrate on the investigation of a single agent by proposing a logic model for its specification and evaluation. All the other agents follow the same model.

We introduce a formal logic model for representing AoS security rules and policies based on a first order language. We explain syntactic and semantic descriptions for this policy base model.

$\mathcal{L}$ is used to represent a sorted first order language with equality, with four disjoint sorts for legitimate object with object constants $O, O_1, O_2, \ldots$, and object variables $o, o_1, o_2, \ldots$, legitimate user with user constants $U, U_1, U_2, \ldots$, and user variables $u, u_1, u_2, \ldots$, and group legitimate user with group user constants $GU, GU_1, GU_2, \ldots$, and user variables $gu, gu_1, gu_2, \ldots$, and legitimate operation $OP, OP_1, OP_2, \ldots$ to represent read, write, update, etc. respectively. $\mathcal{L}$ also includes a ternary predicate symbol request which takes arguments as legitimate user or group legitimate user, legitimate object and legitimate operation respectively; A ternary predicate symbol can which takes arguments as legitimate user or group legitimate user, legitimate object and legitimate operation respectively; A binary predicate symbol $\subseteq$ whose both arguments are group legitimate user; Logical connectives and punctuations: as usual, including equality.

Using language $\mathcal{L}$, a legitimate user $U$ can READ legitimate object $O$ of the datacenter is represented by a ground formula $can(U, O, READ)$. We generally use capital letters for constants and lower case letters for variables. A ground formula is a formula without any variables. A user $U$ requests access to object $O$ of the datacenter by means of $OP$ is represented by a ground formula request($U, O, OP$).

Now we explain the group membership concept. If $GU$ is a group constant representing a specific group users called DIRECTOR. "$U$ is a director" means $U$ is a member of the group $GU$, this can be represented using the formula $U \in GU$. We can also represent inclusion relationships between different user groups such as $GU_1 \subseteq GU_2$ which means that all members of $GU_1$ is also a member of $GU_2$. Furthermore, we can represent constraints among users’ authorizations. Certain access rights are defined by the roles such as a general staff can access general information, a group leader can access his group members personal information, etc. For example, the rule stating that “a director can update confidential file F”, “Alice is a director”, “manager can access confidential file F” can be represented as follows. We use $D$ to represent the group director, $M$ to represent the group manager.

$$\forall u.u \in D \supset can(u, F, UPDATE),$$

$$Alice \in D,$$

$$\forall u.u \in M \supset can(u, F, ACCESS),$$

When access rights are assigned to a group, that means all group members are entitled the access rights the group assigned unless otherwise specified. We define that if a group entitles access to certain object, then all the members of the group can access the same object unless otherwise specified. This is called the inheritance property of authorizations. This can be represented as:

$$\forall u.u \in GU \wedge can(GU, O, OP) \supset can(u, O, OP).$$

Where $u$ represents any member of the group $GU$ and $O$ is a data object of the datacenter that $GU$ can access, $OP$ is the operation performed on $O$.

Here is the formal definition of the security rule base of the AoS by using language $\mathcal{L}$.

**Definition 1:** A security rule base SRB is a quintuple of $(LU, LO, LOP, F, C)$ where $LU$ is a finite set of legitimate users; $LO$ is a finite set of legitimate data objects of the datacenter; $LOP$ is a set of legitimate operations performed on the data; $F$ is a finite set of ground literals and $C$ is a finite set of closed first order formulas.

We define a formula with no free variables as a closed formula. In our formalism, both facts and rule constraints are represented by closed formulas of $\mathcal{L}$. For example, $can(Alice, F, WRITE)$, $Alice \in D$ are facts, $can(Alice, F, READ) \supset can(Amy, F, READ)$, $\forall u.u \in M \supset can(u, F, ACCESS)$, are rule constraints. These rule constraints are viewed as access constraints which should be always satisfied. We refer to a fact as a ground formula, a ground literal, or an atom.

A model of a security rule base is the assignment of a truth value to every formula of the security rule base in such a way that all formulas of the security rule base are satisfied [7]. Formally, we give the following definition.

**Definition 2:** A model of a security rule base $SRB = (LU, LO, LOP, F, C)$ is defined to be a Herbrand model [7] of $LU \cup LO \cup LOP \cup F \cup C$. $SRB$ is said to be consistent
if there exists some model of SRB. The set of all models of SRB is denoted as Models(SRB). A formula $\psi$ is a consequence of SRB, denoted as $SRB \models \psi$, if $LU \cup LO \cup LOP \cup F \cup C \models \psi$. In this case, we also say $\psi$ is satisfied in SRB.

The following examples show how the security rule base work.

**Example 1:** Assume both $U_1$ and $U_2$ are legitimate users, $D$-records is a legitimate data record of the datacenter, and OP is a legitimate operation. Both $U_1$ and $U_2$ belong to a group called STAFF. This group can perform OP on D-records. The constraint states that if someone belongs to a group then he/she inherits the group’s access rights. In our security rule base, this situation can be specified as $SRB = (LU, LO, LOP, F, C)$, where

$$
LU = \{U_1 \in LU, U_2 \in LU\},$

$$LO = \{D\text{-}records \in LO\},$$

$$LOP = \{OP\},$$

$$F = \{U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, D\text{-}records, OP)}\},$$

and

$$C = \{\forall u, g, o, op. u \in g \land \text{can}(g, o, op) \supset \text{can}(u, o, op)\}.$$

It is not difficult to see that facts can($U_1$, D-records, OP) and can($U_2$, D-records, OP) are consequences of SRB, and SRB has a unique model $m$ where:

$$m = \{U_1 \in LU, U_2 \in LU, D\text{-}records \in LO, OP \in LOP, U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, D\text{-}records, OP)}\}.$$

**Example 2:** Suppose one more constraint is added to example 1 as “$U_1$ and $U_2$ are claimed to be conflict of interest, they are not allowed to access the same data object”. In this case, the security rule base is specified as: $SRB = (LU, LO, LOP, F, C)$, where

$$LU = \{U_1 \in LU, U_2 \in LU\},$$

$$LO = \{D\text{-}records \in LO\},$$

$$LOP = \{OP\},$$

$$F = \{U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, D\text{-}records, OP)}\},$$

and

$$C = \{\forall u, g, o, op. u \in g \land \text{can}(g, o, op) \supset \text{can}(u, o, op), \forall o, op. \text{can}(U_1, o, op) \supset \sim \text{can}(U_2, o, op)\}.$$

Two models will yield as:

$$m_1 = \{U_1 \in LU, U_2 \in LU, D\text{-}records \in LO, OP \in LOP, U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, P\text{-}records, OP)}\},$$

or

$$m_2 = \{U_1 \in LU, U_2 \in LU, D\text{-}records \in LO, OP \in LOP, U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, P\text{-}records, OP)}, \text{can}(U_2, D\text{-}records, OP), \sim \text{can}(U_1, D\text{-}records, OP)\}.$$

Now we discuss the evaluation of SRB. When a user $U$ requests access to $D$ of the datacenter by certain operation, the task of the AoS is to evaluate such a request and determine either to grant or deny the request.

For a request $\text{request}(U, O, OP)$, Generally, the AoS will first check its corresponding SRB to find out if $U$, $O$ and OP are legitimate user, data object and operation or not. If yes, it then checks the facts of the SRB, if can($U, O, OP$) presents, $\text{request}(U, O, OP)$ is explicitly granted. Otherwise, it does reasoning about the related facts and rules, calculates the model of the SRB. If can($U, O, OP$) is entailed in the model, then can($U, O, OP$) can be deduced, hence the request is implicitly granted; otherwise, the request is denied.

**Definition 3:** For an access request $\text{request}(U, O, OP)$, the AoS evaluates the $SRB = (LU, LO, LOP, F, C)$ by calculating its model $m$. If can($U, O, OP$) $\in m$, or $SRB \models \text{can}(U, O, OP)$, $\text{request}(U, O, OP)$ is to be granted; otherwise, it is to be denied.

**Example 3:** The SRB is as described as in Example 1. In addition, $U_3$ is also a legitimate user. The access requests are: $\text{request}(U_1, D\text{-}records, OP)$ and $\text{request}(U_3, D\text{-}records, OP)$. In this case, the $SRB = (LU, LO, LOP, F, C)$, where

$$LU = \{U_1 \in LU, U_2 \in LU, U_3 \in LU\},$$

$$LO = \{D\text{-}records \in LO\},$$

$$LOP = \{OP\},$$

$$F = \{U_1 \in \text{STAFF}, U_2 \in \text{STAFF}, \text{can(STAFF, D\text{-}records, OP)}\},$$

and

$$C = \{\forall u, g, o, op. u \in g \land \text{can}(g, o, op) \supset \text{can}(u, o, op)\}.$$

Again, the unique model $m$ is:

$$m = \{U_1 \in LU, U_2 \in LU, U_3 \in LU, P\text{-}records \in LO, OP \in LOP, U_1 \in \text{STAFF,}\ U_2 \in \text{STAFF, can(STAFF, P\text{-}records, OP)}\},$$

$$\text{can}(U_1, D\text{-}records, OP), \text{can}(U_2, D\text{-}records, OP)\}.$$

Obviously, $SRB \models \text{can}(U_1, D\text{-}records, OP)$ is granted; $SRB \models \text{can}(U_3, D\text{-}records, OP)$ does not hold, so $\text{request}(U_3, D\text{-}records, OP)$ is denied.

### 4. The Implementation of AoS

From the above evaluation process, we can see that the major implementation issue is the model generation. For a given SRB, once its set of model is computed, any access request will be answered just by checking the model set.
Since the sets of LU, LO and LOP represent the set of legitimate users, the set of legitimate data objects and the set of legitimate operations. Their truth values are specified by the security agents at the beginning and can be verified easily. In the following discussion, we will skip the verification of these two sets and concentrate on the sets of F and C.

For a security rule base \( SRB = (LU, LO, LOP, F, C) \), generally the constraints in \( C \) may include universally quantified variables\(^2\). From the implementation consideration, we need to ground each constraint containing variables in \( C \) to all of its propositional instances. This technique is often used in the implementation of first order dynamic systems, e.g.[23].

In Example 1, \( C \) contains one constraint:

\[ \forall u, g, o, op. u \in g \land can(g, o, op) \supset can(u, o, op). \]

During implementation, this constraint needs to be replaced by its two ground instances:

\[
\begin{align*}
U_1 & \in STAFF \land can(STAFF, D-records, OP) \\
& \supset can(U_1, D-records, OP), \text{ and} \\
U_2 & \in G \land can(STAFF, D-records, OP) \supset can(U_2, D-records, OP).
\end{align*}
\]

In the rest description, when we refer to the security rule base \( SRB = (LU, LO, LOP, F, C) \), we assume that the set of \( C \) only consists constraints without variable occurrence.

From the implementation point of view, we need to define some additional concepts that will be used in our algorithms. For a security rule base \( SRB = (LU, LO, LOP, F, C) \), an inconsistency is a set of literals whose conjunction is inconsistent with \( C \). A minimal inconsistency is an inconsistency which has no subset that is also an inconsistency.

Let \( L \) be the set of all ground literals of the language defined. To get the set of models of \( SRB = (LU, LO, LOP, F, C) \), first we need to find out the set \( I \) of minimal inconsistencies between \( L \) and \( C \). This can be achieved using an inference engine. Given \( L \) and \( C \), we can use the resolution proof to find out all of the minimum-length proofs which lead to empty clauses, and the required inconsistencies can be directly read off from these proofs.

For instance, let us consider Example 1 and see how one can obtain the set \( I \) of minimal inconsistencies. To simplify the problem, let \( a \) stand for \( U_1 \in STAFF, b \) for \( U_2 \in STAFF, c \) for \( can(T, D-records, OP) \), \( d \) for \( can(U_1, D-records, OP) \) and \( e \) for \( can(U_2, D-records, OP) \). Then this policy base can be viewed as \( SRB = (LU, LO, LOP, F, C) \) where \( F = \{a, b, c\} \) and \( C = \{a \lor c \supset d, b \land c \supset e\} \). Furthermore, \( a \land c \supset d \) is equivalent to \( \neg a \lor \neg c \lor d \) and \( b \land c \supset e \) is equivalent to \( \neg b \lor \neg c \lor e \). Here \( L = \{a, \neg a, b, \neg b, c, \neg c, d, \neg d, e, \neg e\} \).

The resolution proof of Figure 1 shows the procedures to obtain the set \( I = \{\{a, c, \neg d\}, \{b, c, \neg e\}\} \) of minimal inconsistencies.

Once the set \( I \) of minimal inconsistencies between \( L \) and \( C \) is obtained, a model \( m \) of \( SRB \) can be achieved by a maximal subset of \( L \) which contains \( F \) but does not contain any minimal inconsistency.

For the above example, considering the first inconsistency \( \{a, c, \neg d\} \), we get models \( \{a, b, c, d\} \) and \( \{a, b, c, d, e\} \). Taking the second inconsistency \( \{b, c, \neg e\} \) into account, leaves us the only model \( \{a, b, c, d, e\} \). That is,

\[
\{U_1 \in STAFF, U_2 \in STAFF, can(STAFF, D-records, OP),
\quad can(U_1, D-records, OP), can(U_2, D-records, OP)\}\}.
\]

Figure 1 illustrates the steps to the resolution proofs.

![Resolution Proofs](image)

In summary, the algorithm is as follows:

**Model Generator Algorithm**

**Input:** a finite set \( F \) of ground literals, a finite set \( C \) of ground formulas and a finite set \( L \) of ground literals over \( F \cup C \).

**Output:** a finite set of models of \( F \cup C \).

1) Use resolution proof to find the set \( I \) of minimal inconsistencies between \( L \) and \( C \).
2) Find the maximal subset of \( L \) which contains \( F \) but does not contain any inconsistency of \( I \).
3) From the set of all such maximal subsets of \( L \) to form the set of models of \( F \cup C \), i.e., \( Models(F \cup C) \).

In the above algorithm, we achieve step 1 using a theorem prover OTTER [18]. In fact, step 1 can be pre-computed as a separate procedure for finding the set of minimal inconsistencies between \( L \) and \( C \).

**5. Conclusions**

In this paper, we have examined the security issue of the datacenter in cloud computing environment. We proposed
a structure to authenticate the users and to control their accesses in order to protect the datacenter from malicious attempt. We have sketched a framework for the authentication process and introduced a detailed formal approach for the access control mechanism. We investigated a logic approach for representing authorization rules and evaluating user’s access request.

This paper is the extension of our previous work [1] where legitimate operation has been added and two binary predicates request and can have been augmented to ternary predicates in order to better capture the essence of the security rules and policies. And also the implementation issue has been considered and discussed. The resolution proof is employed into the implementation process. However, more detailed access rights, different operations on data object need to be investigated. This part will be investigated in our future work.

References