ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

An Auditable Attribute Based Access Control Mechanism in Openstack Cloud Environment

Sachin Patel, Prof. Yagnik Rathod

SardarVallabhbhai Patel Institute of Technology, Vasad, India

Assistant Professor, SardarVallabhbhai Patel Institute of Technology, Vasad, India

ABSTRACTP: Access control activity is often considered as limiting the activities of legitimate users to the system and protecting the system from un-authorized usage of the system. There are various access control mechanism available and widely used from 1960s to till date, few of them are very popular and widely used access control models. They are Disctionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC) and Attribute Base Access Control (ABAC). Previous three models have been widely accepted and used. On the other hand ABAC is relatively new and lot of work has been done and formal models are developed. An ABAC allows or deny the access to the resource. Example subject attributes are name, designation, organization, project etc. where as example object attributes are size, type, name etc. In addition to the attributes we can also enforce environment conditions such as time and location. But the problem with the attribute base access control is "auditing". As the access control decisions are made only based on the attributes of the subject and resources we don't have the luxury to express each and every attributes for auditing of the subject and object as these attributes may not be known to the auditor of the system. Here in this paper we will discuss the possible solution for the same in open stack cloud environment.

KEYWORDS: Attribute Based Access Control, Auditing, Cloud Computing, Openstack

I. INTRODUCTION

Access control is often coupled with effective auditing mechanism[7]. And to provide effective auditing of the system it is necessary to provide effective authentication and authorization of the system. Access control often considers that authentication of the user or subject is done prior to the request made by the user. Access control deals with the authorization part. Here we consider system as union of both subjects and objects, subject can be a process initiated by the user or user itself, and resource can be database, files or virtual resources and machines. There is a distinction between policy and mechanism [7].

First popular model is Discretionary Access Control Model. Discretionary access control models are similar like early UNIX file system, where the owner of the file decides who can access the file and in which mode they can access. Discretionary policies govern the access of users to the information on the basics of user's identity and authorizations or rules that specify, for each user and each object in the system, the access modes like read, write and execute that user is allowed to do on that object. Each request of a user to access on object is checked against the specified authorizations. If there exists an authorization stating that the user can access the object in specific mode, the access is granted, otherwise it is denied[7].The problem with DAC is it doesn't provide real assurance of discrimination of information.

Second popular model is Mandatory Access Control model. Mandatory access control policies address the problem of discrimination of information by attaching security labels to both users and resources in system. Mandatory policies govern access on the basics of the classification of subject and objects in the system. Each user and each object in the system is assigned a security level. The security level connected to object reflects importance of the object. The security level associated with the user is called clearance and it reflects the trustworthiness in the system[7]. There are basically two operations read down and write up[7]. The problem with MAC is it only provides one directional informational flow.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

The popular and widely used access control mechanism is Role Based Access Control Mechanism or RBAC. Role is main idea behind the RBAC models. System administrator creates roles according to the job function performed in a organization and grants permissions to those roles, and then assign these roles to the users of the system on basis of their responsibility in an organization.It is easy to assign permission to roles and then assigning roles to users rather than creating separate set of permissions to each user in system.[5] Roles of the user can be changed when needed as per the current requirement of the organization. RBAC is policy neutral means that it can be implemented by DAC or MAC. RBAC supports various access control principles like separation of duty, least privilege and data abstraction. The problem with the RBAC is role explosion and role engineering.

The newer and current Access Control Mechanism is ABAC. Basic idea behind ABAC is to match the attributes of the subjects and objects to make the access decision. If the attributes of subject and object match the access is allowed and if attributes does not match access to object is denied. Example subject attributes can be name, employee no. , designation , or department, on the other hand object attributes can be name, size, creation time etc. we can have as many attributes as system needs. In addition to this ABAC supports mutable attributes[1]. We can also implement environment condition to support decision making system. Example environment attributes can be time, location of user and ip address of the user. ABAC provides us the fined grained access control to ensure maximum protection to the system. ABAC system also supports least privilege and separation of duty principles of access control. ABAC is scalable and easy to implement for the system.

Problem with the ABAC is the auditing of the system is very difficult compared to RBAC. It is essential to have efficient auditing mechanism. In following sections we will discuss about the auditing process in RBAC and ABAC.

II. RELATED WORK AND PROBLEM STATEMENT

In RBAC When a user invokes its role to perform the task the session is started and maintained till user complete its task[5]. These sessions can be used for auditing of the systems. Further the auditor can use the roles used by the user and the sessions which are maintained for the user this makes the auditing extremely easy in RBAC systems.

On the other side there are no sessions maintained when user start and stop their task. And the user can have many attributes unlike RBAC where users have roles only. If organization of the system decides to move their current system from RBAC to ABAC the only way is to consider the role as attributes in such cases the advantage of auditing is lost[3]. As the ABAC systems purely work on attributes only we don't have that luxury to express each and every attribute of the user for the auditing process[3].

We are here considering openstack as a cloud environment. There are seven basic services provided by openstackare nova, keystone, glance, swift, horizon, cinder and neutron. Nova is storage service, keystone is authentication service, glance is image service, swift is object storage service, horizon is dashboard, cinder is block services and neutron is network service. Each user can use each of these services or combination of this.

As we have stated earlier ABAC only works on attributes so when user wants to use any of these service their attributes will be considered and decision is revelled. So for auditing we need to use the attributes in order to track the user.

For example user of the system is Alice and his attributes are name, organization, designation, security_clerance, department etc. and Alice uses the openstack nova service for computation purpose of his task, alice uses the virtual machine named VM1 and its attributes are name, size, security_clerance etc. so when auditor wants to audit the usage of Alice he needs to have list of all attributes of Alice and all attributes of the resource used by Alice. Here we only listed the nova services only but consider that Alice is using all services provided by opnstack and he uses more than one resource of each service. Here the problem becomes more complex and difficult.

Thus the auditing of the ABAC systems becomes much difficult and complicated. Implementation of ABAC is easy and it provides fine grained access control mechanism. But auditing remains problem in such systems. When auditing is not done properly and efficiently it is difficult to track the activities of the users.

In following section we will discuss the possible solution for the problem especially in openstack cloud environment.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

III. PROPOSED SOLUTION

One way is to use the log files provided by the openstack services. Openstack creates and maintains the log files for each services. Such log files are nova.log, keystone.log, glance.log, horizon.log and cinder.log. These log files can be found at var/log/…. .log. We can use this individual log files for auditing purpose.
We will use these log files and will create the SQL database. For storing the log files in database we are using JDBC drivers. Using JDBC drivers we can enter the values of log file into database. Below is the proposed scheme.

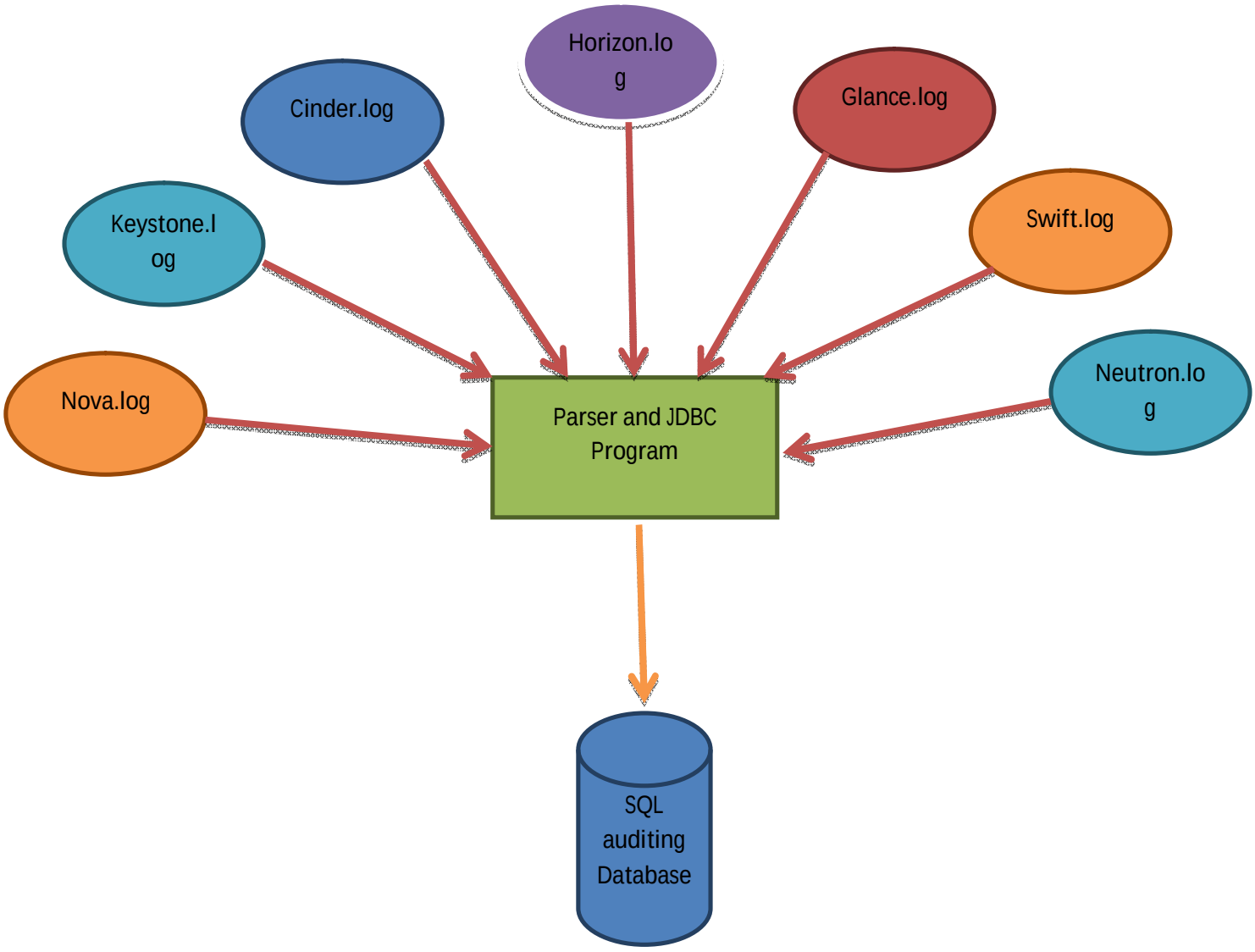

Figure 1: Proposed System

So now when user make any request or access any resource in openstack cloud environment the log files keep the record of the individual services. In the proposed system we have collected the information from all the log files and we

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

store them into central SQL database. So, when auditing needs to be done the auditor can directly access the database and fetch the information from it. For example recall our previous example of Alice where he uses the all openstack services. Now instead of checking individual log files the auditor only needs to check the auditing database only. It will help to keep the auditing easy and simple.

IV. PROPOSED ALGORITHM

Here we propose an algorithm for inserting log file data into MySQL database. We use JDBC driver for the same.

1. Set JDBC driver credentials
2. Set database credentials
3. Register JDBC drivers
4. Open a JDBC connection
5. Execute a query
6. Read log files
   while ((sCurrentLine = br.readLine()) != null) {
   //Check for empty lines
   if(sCurrentLine.length() > 30)
   insertIntoDatabase(sCurrentLine);
   }
7. String regEx = "\\s+|,\\s*|\\:\\s*"
8. String [] content = sCurrentLine.split(regEx);fields
9. int i = m (m= no. fields)
   while(i != content.length){
   message = message + " " + content[i];
   i++;
   }
10. sql = "INSERT INTO tablename " +
    "VALUES()”

V. RESULTS

As we have created a single database which includes all the data from all log files. So auditor has to only access a single database not all log files. Here auditor can fire queries as needed for auditing of the system. Below figure 2 shows the database table for the auditing database for openstack cloud.
We have perfomed various tasks such as auditing of the users as well as auditing of the resources manually. We found that we have decreased the time tremendously by our system. In former system we have to go through all log files and we have to check the each and every entry. Moreover we found that log files are updated by the system at each second this makes task even more difficult.
Where as in our system we found that the same auditing process can be done very easily by just executing an query.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

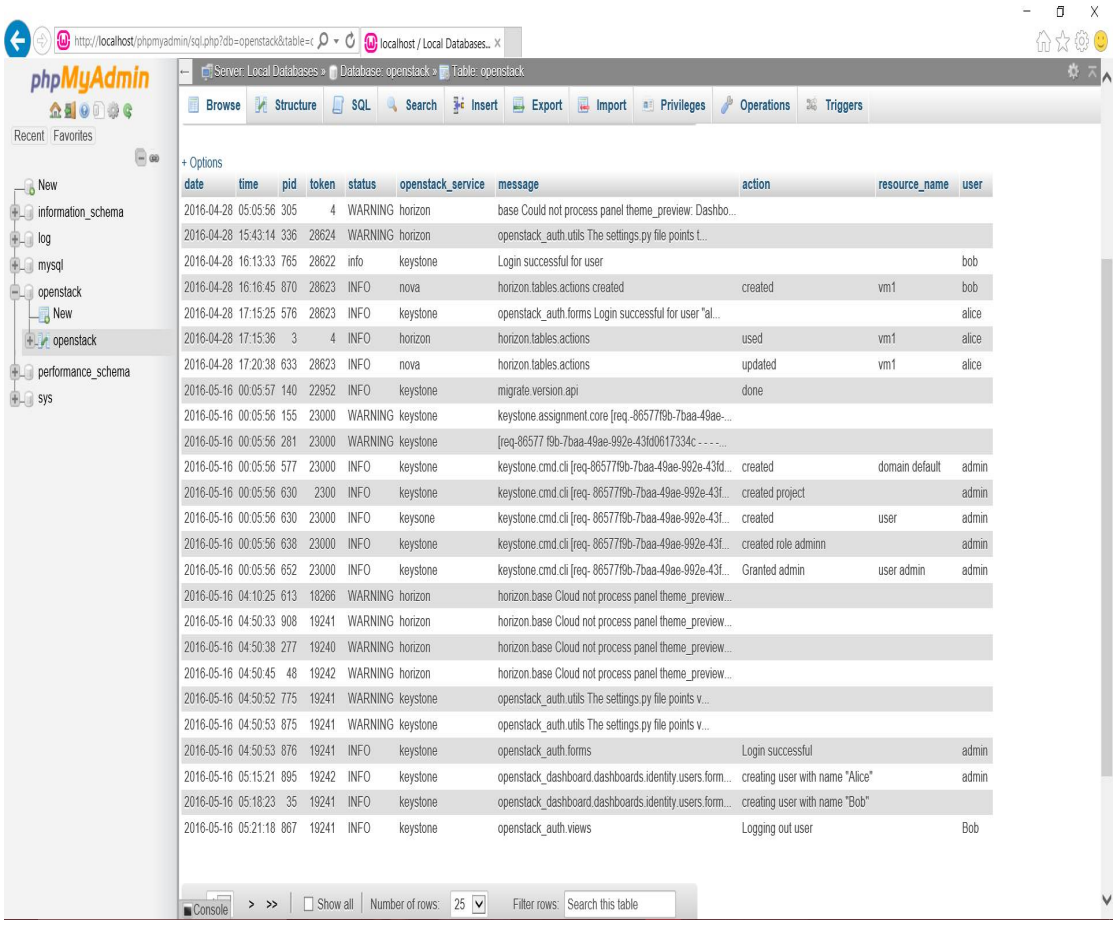

Figure 2: Openstack Database Table

Below graph shows the improvement in result.

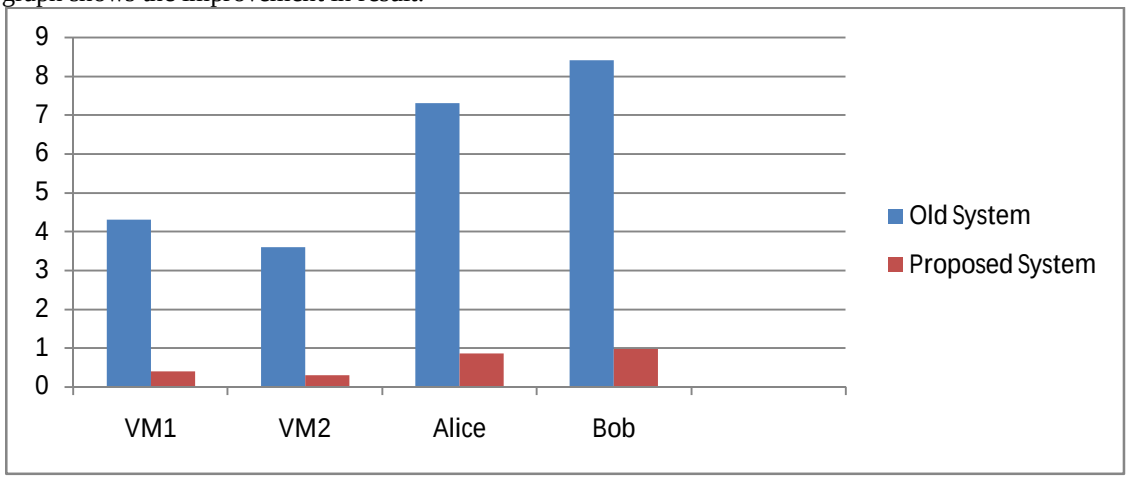

Table 1. Results

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

% Improvement = $\frac{total\ old\ time - total\ new\ time}{total\ old\ time}$ (100)

$= \frac{23.8\ min - 1.7 min}{23.8\ min}$ (100)

= 92.8%

VI.CONCLUSION AND FUTURE WORK

The attribute based control access control mechanism is easy to implement and flexible, it provides fine grained access control over resources. The auditing is difficult but using our proposed system auditing is made easy and efforts for auditing will be less. We have achived 92.8% improvement in existing auditing process. In future we want to work on various parsing mechanism for converting log files into database records.

REFERENCES

[1] Jin, Xin, Ram Krishnan, and Ravi S. Sandhu. "A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC." *DBSec* 12 (2012): 41-55.
[2] Bijon, Khalid Zaman, Ram Krishman, and Ravi Sandhu. "Constraints specification in attribute based access control." *Science* 2.3 (2013): 131-144.
[3] Hu, Vincent C., et al. "Guide to attribute based access control (ABAC) definition and considerations (draft)." *NIST Special Publication* 800 (2013): 162.
[4] Jin, Xin, Ram Krishnan, and Ravi Sandhu. "Role and attribute based collaborative administration of intra-tenant cloud IaaS." *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*. IEEE, 2014
[5] Sandhu, Ravi S., et al. "Role-based access control models." *Computer* 2 (1996): 38-47.
[6] Ahn, Gail-Joon, and Ravi Sandhu. "Role-based authorization constraints specification." *ACM Transactions on Information and System Security (TISSEC)* 3.4 (2000): 207-226.
[7] Sandhu, Ravi S., and PierangelaSamarati. "Access control: principle and practice." *Communications Magazine, IEEE* 32.9 (1994): 40-48.
[8] Onankunju, Bibin K. "Access Control in Cloud Computing." *International Journal of Scientific and Research Publications* 3.9 (2013)
[9] Sen, Jaydip. "Security and privacy issues in cloud computing." *Architectures and Protocols for Secure Information Technology Infrastructures* (2013): 1-45
[10] Coyne, Ed, and Timothy R. Weil. "ABAC and RBAC: Scalable, flexible, and auditable access management." *IT Professional* 3 (2013): 14-16