International Journal of Security and Its Application
Vol.11, No.1 (2017), pp.245-256
http://dx.doi.org/10.14257/ijsia.2017.11.1.20

Enhancement of Cloud Authorization System Using Predicate Logic

Sandeep Saxena[1], Goutam Sanyal[2] and Shashank Srivastava[3]

Asst. Prof.,[1]Krishna Engineering College,Ghaziabad, India
Reseach Scholar, [1]National Institute of Technology, Durgapur, India
[3]MNNIT Allahabad, India
saxena.s.in@ieee.org, Shashank12march@gmail.com

Abstract

Cloud computing is an emerging technology that facilitates companies in the IT sector and Education Sector. Authorization is a major security concern in any organization. The small organization is not interested to implement bulky access controls that are utilized a large amount of bandwidth and cost.In this paper, we had analyzed the problem of authorization computation complexity and so we give an efficient light weighted authorization method. Authorization process involves number of rules validation before providing access to resources. We propose an Authorization process that is less complex then exiting Authorization access control methods, in which we use predicate logic rule-set, which reduce the policy misinterpretations and policy misconfigurations. In which we enable the architecture with fixed policy rules and user-defined policy rules. We did target the small organization for design this framework. In this paper we consider case study of educational system and in which we find better results than other authorization systems.

Keywords*: Authorization, Predicate Logic, Authorization Rule Sets, Cloud Computing, Authorization Framework.*

1. Introduction

Cloud computing is a pool of resources internally connected and shared with multiple parties; resources are easily provisioning and de-provisioning to the clients. Cloud computing is completely based on virtualization technology and authorization is major security concern as it growing worldwide.

In the current developing scenario, most of the organization interested to build own cloud environment that is called 'Private Cloud.' Rather than taking the service from a third party, a big organization interested in building their infrastructure, software or platform cloud that require a large budget and a lot of skilled resources. Due to this reason, cloud computing technology is growing rapidly and improving to accommodate most of the organization requirements. While there are a variety of benefits to adopting cloud computing, there are also many challenges facing cloud computing are security and privacy issues such as authentication, authorization, and accounting. So we need a secure access control for cloud computing technology.

This paper describes an authorization framework and generic architecture for our case study, which is much closely related to any academic organization and that has to be interested to build own cloud environment within the organization. Any organization have their own data and interested to be make it confidential and secure so most of the organization do not have trust in third-party cloud service provider and interested to develop own cloud infrastructure within organization and maintain and run this cloud service with minimum human effort. They do not need highly complex, and bulky security mechanism because they need to access all resources within the organization and

ISSN: 1738-9976 IJSIA

by the internal employee only. We already proposed authentication protocol for same infrastructure **[1]** and in this paper we proposed an authorization framework that will manage and maintain with minimum human effort and very much cost effective for any academic organization. This is authorization process consumes less memory and time during a large amount of client authorization process.

2. Related Work

Our proposed authorization framework is quite similar to previous research work done in the same domain. In 2001 jajodia *et al*. [2] proposed a flexible authorization Framework (FAF) called FAF to enforce various access control policies within an integrated data store. Similar to this paper other web service access control, involves logical languages [3], [4], [5] and context-aware models [6]. Some other access control models use eXtensible Access Control Markup Language (XACML) [7] and Security Assertions Markup Language (SAML), but XACML are used for the access control mechanism. XML knowledge is requiring writing rules in XACML, because all access control policies must be expressed in XML only. Our model does not require these constraints for user/administrator. In 2006, Weider D. Yu *et al*. [8] proposed a new language ARSL to overcome the previous issues in implementing authorization in Web Services. David W. Chadwick *et al*.[9] proposed a privacy-preserving authorisation system in which user can set their privacy policies and then enforce them so that unauthorized access is not allowed to their data. In this system, they designed Master PDP (policy decision point) which resolves policy conflict and user_PDP to make authorization decision that will enforce by PEP (Policy enforcement point).

Similar work is done in Grid environment by Mustafa Kaiiali *et al*. [10] proposed Grid Authorization graph techniques in which they use Hierarchical Clustering Mechanism (HCM) to handle repetitions in checking security rules but HCM is not entirely free from repetitions and cannot handle OR-based security policies then they proposed Grid Authorization Graph to overcome HCM limitation in same paper. Umer Khalid *et al*. [11] introduced an architecture for secure and privacy-enhanced authentication and authorization architecture in which they use IDMS (identity management system), token mechanism and certificate authority to authenticate the user. For authorization, token forwarded from the authentication server to the authorization server. Authorization server received an encrypted access token and decrypt the token using symmetric to get user ID then authorization server checks for policies against user ID. Based on policies using policy combiner algorithm that result in the permit, deny or undefined.

In 2013 Amal Alsubaih et, al. [12] proposed an Authorization as a Service for Cloud Environment, in which they use prolog-style stratified logic programming rules to define the authorization policies which is similar with FAF. In this paper they designed permission formation architecture that includes information about Role store, Role Hierarchy, Permissions, Business Task, *etc*. they included consistency check Completeness Check and conflict check. Nelson Mimura Gonzalez *et al*. [13] proposed a framework for authentication and authorization based on credentials in cloud computing. In this paper, they established a relationship between entities, attribute, identities, and credentials. Based on credentials they stated that what action an object can perform in cloud environment and credentials can also define rules and policies that an entity has to follow within the context of cloud, which includes financial obligation, legal aspect, and operational directives.

In 2014 Ehtesham Zahoor *et al*. [14] proposed a CATT Framework includes trust and temporal aspects. This model can handle different qualitative, quantitative, and periodicity based temporal constraints. They are also presented architecture for authorization policy evaluation using DECReasoner Tool.Various number of authorization systems are

available in current scenario but time to compile authorization rules and ambiguity is still a big problem. Because cloud is dealing with very large number of users per second so still we need a new authorization method and architecture for minimizing these two problems. So we did start our work to resolve this problem and developed a new framework for authorization system.

3. Proposed Work

Authorization is a process by which only authorized user/client will access the resources. The resource may be files/ machine / software, *etc*. we assume that cloud environment is secure and only authenticated user/client will send an access request to the server. Within which authorization server will check the authorization rule set to verify that access request have sent by an authorized user/client only. In this paper, we proposed authorization framework in which user/client privacy and data are hidden from other users. We provide the framework that resolves policy inconsistencies and conflicts within authorization rule sets. The basic idea of our solution is given in Figure 1. In which we define the Authorization Policy Store, contains the following information.

Role: user role by which access request is generated.

Object: Object is representing user that associated with the particular role.

Resources: resources may be file/ machine/ software, *etc*. here we select the file as resources.

Role Hierarchy (optional): role hierarchy information is stored in the database but it is optional because some organization is not required to define it.

General Policy: general policy defined by the organization based on role or role hierarchy.

User Defined Rules: these rules are defined by resources owner. The owner is not have right to give permission to another user /client. It includes date and time of start and end of this access permission.

Conflicting permission check: it is based on "conflict of interest" principals.

All set of authorization rules: it will include all rules sets defined by the general policy scheme and user-defined rule set.

In this paper, we use predicate logic (so called first order logic) to defined rules for authorization.

Why we select predicate logic?

1. It defined rules without policy misinterpretation (no ambiguities).
2. It defined rules without policy misconfigurations.
3. Predicate logic is abstract, flexible and extensible.
4. Its enforcements mechanism with proofs.

In above authorization framework, all rules are stored in authorization policy store in the form of predicate logic expressions. The compiler will compile all rules and store in compiled authorization rule set. When user/client send access request, then it will check and validate in a layer of "compiled authorization rule set". User (resources owner) will grant permission to another user for access resources that created (OR having authority) by that user.

Table 1. Grammar

a. Formal Grammar for Predicate Logic (as per our architecture)

Variable: $\mathbf{v = x|y|z \ldots}$

Constant: $\boldsymbol{c = r_1|r2|r3\ldots\ldots|o1|o2|o3\ldots}$

Term : $\boldsymbol{t = v|c}$

Predicates $\mathbf{P = A|O|R|HR|F|OW}$

$\mathbf{A = Access, O = Objects, R = Resources, HR = HR\ dept,}$

$\mathbf{F = Finance\ Dept/Account\ Dept}$

$\mathbf{OW = Owner}$ $\boldsymbol{Atom: = Pt1, t2, t3\ldots\ldots tn\ , where\ n\ is\ the\ arity\ of\ P.}$

Expression: $\varphi = \boldsymbol{Atom}| \neg \varphi \mid \varphi \wedge \varphi \mid \varphi \boldsymbol{V} \varphi \mid (\varphi \rightarrow \varphi) \mid \varphi \leftrightarrow \varphi \mid \forall \varphi \mid \exists \varphi$

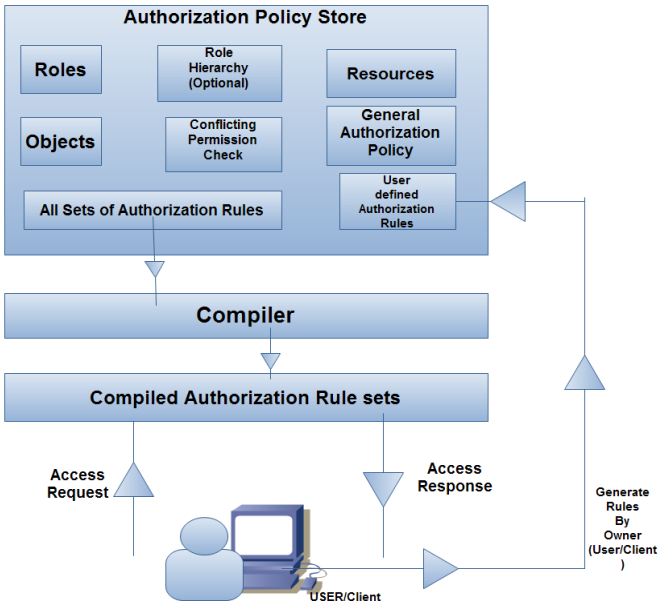

Figure 1. Authorization Framework

Table 2. Formal Notations

Oo_1 : $Object\ is\ o_1$
$O_{(o1 \vee o2)}$: $Object\ o_1\ OR\ o_2$
A_{r1} : $Access\ resource\ r_1$
$A_{o1,r1}$: $Access\ resource\ r_1\ by\ object\ o_1$
$A_{o1,o2}$: $o_1\ access\ all\ resources\ of\ o_2$

b. **Derivation of Authorization Rules**

Hierarchical system

In the below mention tree structure represent objects and resources based on access permission. Here $o_1,o_2,o_3,o_4,o_5,o_6,o_7$ represent objects and r_1,r_2,r_3,r_4,r_5 represent resources.Here R_1= r_1, r_4 and R_2=r_1, r_4, r_5.

Based on below hierarchical structure If o4 access R1 (r1, r4) then o2 can also access R1 and o2 have the right to access the o4 object.

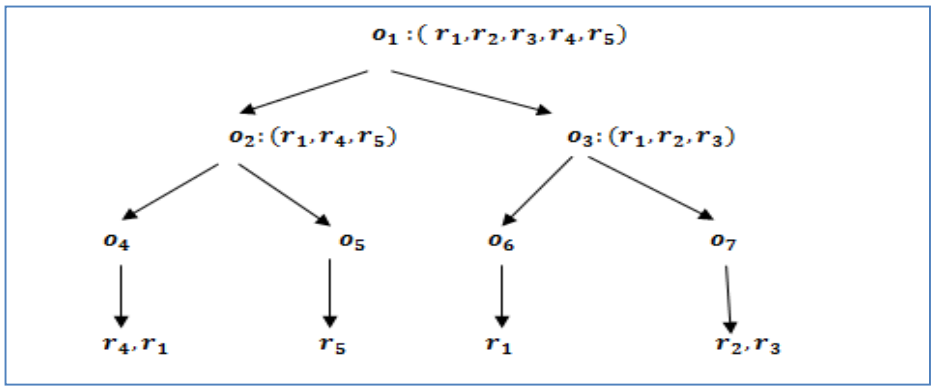

Figure 2. Hierarchical Structure of Object and Resources based on Access Permission

$A_{o2,o4} \Lambda A_{o4,R1} \rightarrow A_{o2,R1}$ (Transitive Rule)
If o5 access r5 then o2 can also access r5 and o2 have the right to access the o5 object.
$A_{o2,o5} \Lambda A_{o5,r5} \rightarrow A_{o2,r5}$ (Transitive Rule)
If o2 having right to access R1 and r5, then we can write
$A_{o2,R1} \Lambda A_{o2,r5} \rightarrow A_{o2,R2}$ (Summation Rule)
Where R_2= $R_1 \Lambda r_5$
Similarly for o3:
$A_{o3,r1} \Lambda A_{o3,R3} \rightarrow A_{o3,R4}$ (Summation Rule)
Where R_3=r_2, r_3, and R_4= $R_3 \Lambda r_1$
Similarly for o1:
$A_{o1,R2} \Lambda A_{o1,R4} \rightarrow A_{o1,R5}$ (Summation Rule)
Where R_5=$R_2 \Lambda R_4$ (R_5=r_1, r_2, r_3, r_4, r_5)

In programming implementation we need a algorithm to calculate access true or NOT true. We use following algorithm for our implemtation model. In Algorithm 2, we use it for hierarchical structure to find that object is having authority to access resource yes/no.

In our arcitecture we store each object position in our database and fetch that information when we want to calculate result value.

Algorithm 1

Input: Name of object and name of resource r
Output: true /NOT true in result variable

Algo:

```
2.1     Let result =NOT True;
2.2     Fetch object position from database.
2.3     Consider object position as root node.
2.4       Start tree traversal from root to leafs (lowest level).
2.5           If (node = r)
2.6           result=true;            // break the loop//
2.7        End traversal;
2.8     return result;
```

NOTE:

In our model each and every object is have their fixed position in hierarchical tree structure (as you know, we consider architecture of educational institution). We store each object position in our database in table format structure.

We consider object position as root postion because if any sub node having access right to resource then only root have right to access that resource. We start tree travarsal from object postion in step 2.4. when any node or leaf node having value **r** (in step 2.5)we stop travarsal and change value of result in step 2.6.

c. Optimization of Rule Sets

This is one of the essential features of our proposed work in which we will optimize our rule set that on the basis of minimum checking or scanning of rules we will check the authorization of any user.

Table 3. Object and their Access Resources

Objects	Access Resources
o_1	r_2, r_3, r_4
o_2	r_1, r_4
o_3	r_3
o_4	r_4
. . .	. . .

We take above mention table 3 as example to explain this feature.

For resources r_1:

Rule 1: If object is o_2 then access r_1

$Oo_2 \rightarrow A\ o_2, r1$

Rule 2: If object is o1 then NOT access r1

$Oo_1 \rightarrow \neg A\ o_1, r_1$

Rule 3: If object is o3 then NOT access r1
$Oo_3 \rightarrow \neg A\ o_3, r_1$
Rule 4: If object is o4 then NOT access r1
$Oo_4 \rightarrow \neg A\ o_4, r_1$
Rule Optimization: We can integrate these four rules into single rule:-
No object can access r_1 except access by o_2 OR
For all x in set of O except $x=o_2$ cannot access resources r_1
$\forall\ x\ ((x \in O \Lambda\ x \neq o2) \rightarrow \neg A\ x, r1\ )$

d. Syntax Tree of Authorization Rules

In this part, we will draw syntax tree for the purpose of designing compiler for this predicate logic formulation.

For Example:

$\forall\ x\ ((x \in O \Lambda\ x \neq o_2) \rightarrow \neg A\ x, r_1)$

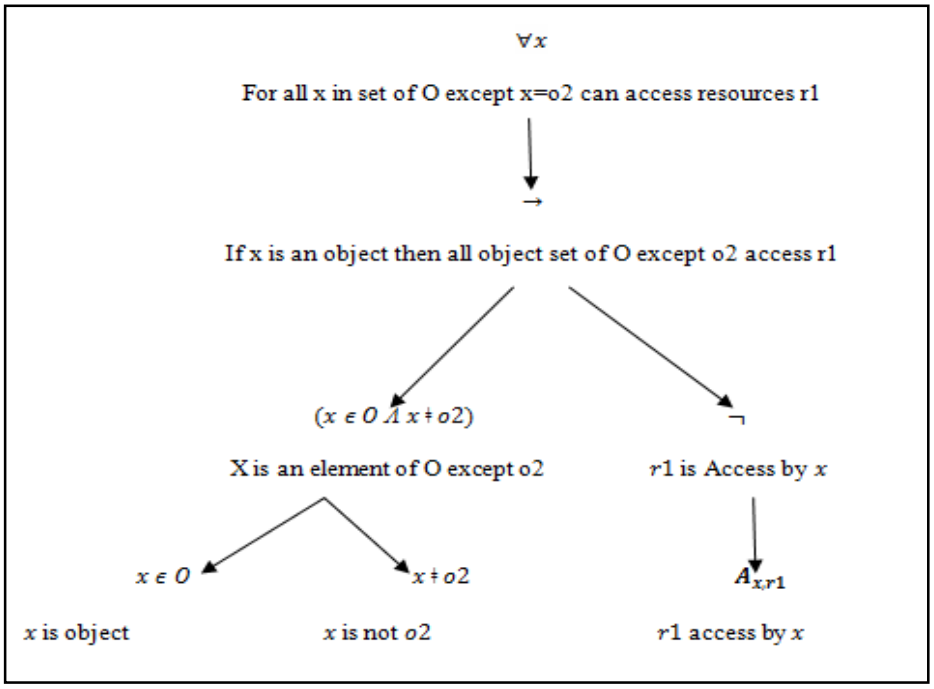

Figure 3. Syntax Tree for "$\forall x\ ((x \in O \Lambda\ x \neq o2) \rightarrow \neg A\ x, r1)$"

4. A Case Study of Feasibility of Proposed Work

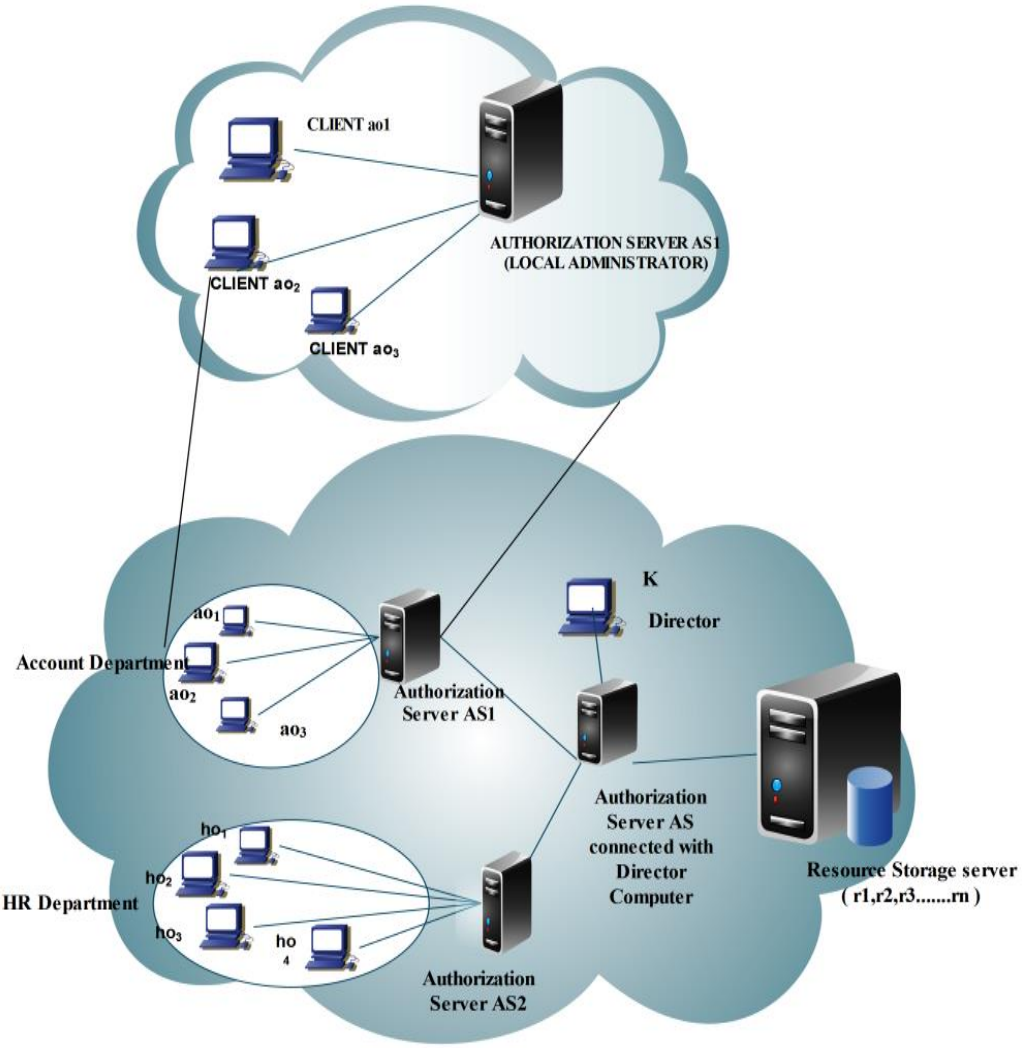

Figure 4. Generic Architecture for Proposed Authorization Framework

We consider an educational system in which multiple departments are running with their responsibilities. In an education institute consist of various departments like account department, HR department, registrar office and various branch wise department. Here we consider two department HR department and account department as sample execution of proposed authorization system as shown in given Figure 4.

Following Terms are used to define this framework:

Resources: $r_i \mid i \in \{1,2,3 \ldots n\}$
Resource Sets: $R_i \mid i \in \{1,2,3 \ldots n\}$
Authorization rules: $ar_i \mid i \in \{1,2,3 \ldots n\}$
Authorization Rule Sets: $AR_i \mid i \in \{1,2,3 \ldots n\}$
Objects: $o_i \mid i \in \{1,2,3 \ldots n\}$
Object Sets: $O_i \mid i \in \{1,2,3 \ldots n\}$
Authorization Servers: $AS_i \mid i \in \{1,2,3 \ldots n\}$
Owner: $OW_i \mid i \in \{1,2,3 \ldots n\}$

Here we consider the resources as files created by department employee's (view as objects). In this framework, we will find some roles/responsibilities/designations in the department as objects and different file records as resources. We are having some responsibilities of account department like account manager, credit and collection manager, Payroll Manager, Tax accounting.

Table 4. Object with their Authorized Resources

Object/Responsibility/role (O)	**Access Resources**
Account Manager (o_1) OR **AM**	Student Records(r_1), Employee Records(r_2), Employee Monthly Salary(r_3), *etc*. (**all Documents**)
Credit and Collection Manager(o_2) OR **CCM**	Students Records (r_1), Student Fee Collection (r_5), Govt. And Other organization funds(r_8)
Payroll Manager (o_3) OR **PM**	Employee Records(r_2), Employee Monthly salary Calculated (for each month)(r_4)
Tax Accounting (o_4) OR **TA**	Employee Monthly Salary (r_3), Employee Monthly salary Calculated (for each month)(r_4), College other Expenses(r_6), Purchasing and selling(r_7)

Above mention table explains the Authorized Access permission for each object. Each resource associated with their owner, which create the resource or having the right to give permission to access for other objects. We create this table based on assumptions, and it will vary from each organization.

Table 5. Resources with their Owner

Resources (Files) (R)	**Owner of Resource (OW)**
Student Records (r_1) OR **Stu_R**	Account Manager (o_1)
Employee Records(r_2) OR **Emp_R**	Account Manager (o_1)
Employee Monthly Salary (r_3) OR **EMS**	Account Manager (o_1)
Employee Monthly Salary Calculated(for each month) (r_4) OR **EMS_ Cal**	Payroll Manager (o_3), Account Manager (o_1)
Students Fee Collection(r_5) OR **Stu_FC**	Credit and Collection Manager(o_2), Account Manager (o_1)
College Other Expenses (r_6) OR **COE**	Account Manager (o_1)
Purchasing and selling (r_7) OR **PAS**	Account Manager (o_1)
Govt and other Organization funds (r_8) OR **GAOF**	Account Manager (o_1)

In above mention, table describes the ownership of each resource. In which account manager having ownership of each resource because account manager is a higher level in account department.

After describing and formulating these tables, we will start to write authorization rules for each object. Rules are associated with each resource because request always generated for resource and be checked at Resource Storage server.

5. Comparison with Other Authorization Methods

In our proposed model, there is no requirement of the knowledge of any programming language. Where as XACML based authorization model requore knowledge of XML Language. Using our model we reduce the cost of education or small organization because if they hire a third party for authorization purpose then they need to pay a considerable amount for this service. The predicate is very simple and

understandable logic language that has very simple syntax to write any new authorization rules. It has no ambiguity and consistent style. It is flexible and easily extensible. Unlike the authorization provided by ".NET framework" this framework decouples the authorization rules for the corresponding organization. Authorization rules and code for implementing rules are same.

6. Result and Analysis

In this section, we evaluated scalability and performance of proposed framework and the developed compiler for parsing authorization rules set. The following graph shows no of functions and time in milliseconds in a single access rule. We draw this Figure 5 using several input files. Each file had a single rule with number of constraints. The above diagram shows that as well as we increase the no of functions time increases gradually and maintain the performance linearly. Its shows our framework is easily scalable and retains the scalability with minor changes in time to execute the functions.

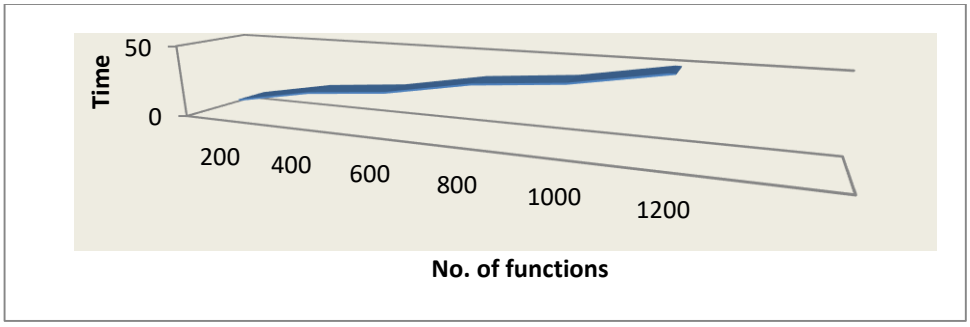

Figure 5. Time Vs No. Of Functions

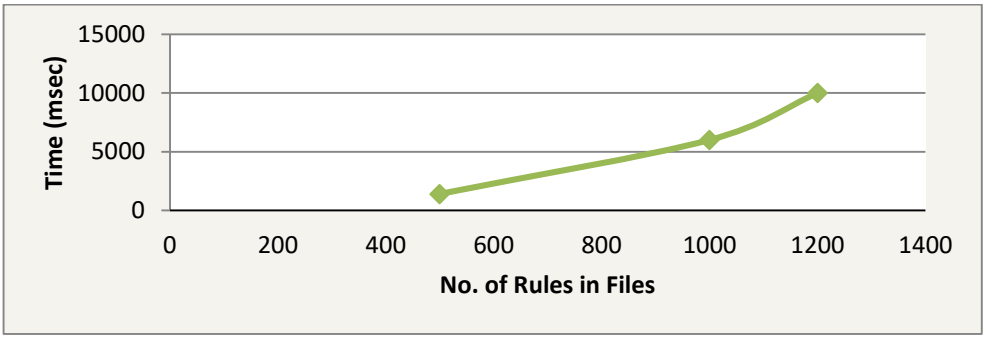

Figure 6. No. Of Rules Versus Time

The second graph (Figure 6) we draw to see that if no of rules increases then what effect on our framework. This graph shows the tradeoffs between number of rules in files and time in msec. Here we can see that as we increase the number of rules, the rules checking time increase gradually but it is better as compared to other framework where time increase exponentially.

After theses two we will draw a graph (Figure 7) between no. Of predicate rules executed versus time require for execution in our framework.

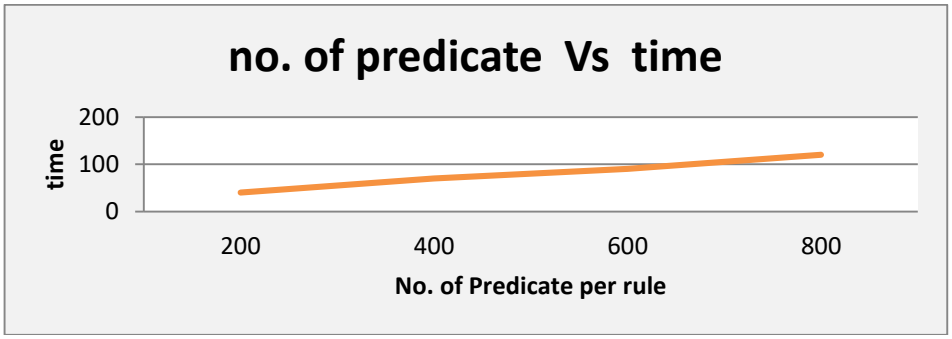

Figure 7. No. of Predicate Versus Time Graph

The data obtained from above experiments shown in the Figure 7. The time increasing as well as no. of predicate execute for each rule is increasing.

7. Conclusion

Authorization is a significant security issue in cloud environments, and customers are ready to pay big amount for this service. But this is a big headache for a small organization that moves to cloud service. By using this framework, they can manage authorization process with minimum effort and cost. The main features of our proposed framework are authorization rules consistent, complete and conflict check. Implementation of our framework in any organization is smooth and comfortable to maintain the authorization process. The time to generate code in a high level language is not a major overhead, and other authorization methods adopt this framework cab based on attribute based, role-based, content based, *etc*.

References

[1] Sandeep Saxena, Goutam Sanyal, Shashank Srivastava, "Mutual Authentication Protocol Using Identity-Based Shared Secret Key in Cloud Environments", IEEE ICRAIE, 2014.
[2] Jajodia, S., Samarati, P., Sapino, M. L., & Subrahmanian, V. S. (2001).Flexible support for multiple access control policies. ACM Transactions on Database Systems (TODS), 26(2), 214-260.
[3] M. Coetzee, J. H. P. Eloff, "A Logic-Based Access Control Approach for Web Services", Proceeding of the 2004 ISSA Information Security Conference.
[4] S. Jajodia, P. Samarati, V.S. Subrahmanian, " A logical language for expressing authorizations", Proceedings of the 1997 Symposium on Security and Privacy, 1997; 31-42.
[5] E. G. Sirer, K. Wang, "An Access Control Language for Web Services", SACMAT '02.
[6] R. Bhatti, E. Bertino, E. Ghafoor, "A Trust-Based Context-Aware Access Control Model for Web-Services",Proceedings of the IEEE International Conference on Web Services (ICWS'04).
[7] eXtensible Access Control Markup Language,www.oasis-open.org/committees/ xacml/repository/cs-xacmlspecification-1.1.pdf
[8] Weider D. Yu, Ellora Nayak," ARSL: A Language for Authorization Rule Specification in Software Security" 11th IEEE Symposium on Computers and Communications (ISCC'06)
[9] David W. Chadwick, Kaniz Fatema,"A privacy preserving authorisation system for the cloud" Journal of Computer and System Sciences, Elsevier,2012, pp. 1359-1373.
[10] Mustafa Kaiiali, Rajeev Wankar, C.R. Rao, Arun Agrawal, Raj Kumar Buyya," Grid Authorization Graph" Elsevier Journal , Future Generation Computer Systems, pp. 1909-1918
[11] Umer Khalid, Abdul Ghafoor, Misbah Irum, Muhammad Awais Shibli, " Cloud based Secure and Privacy Enhanced Authentication & Authorization Protocol", 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013, Elsevier, pp. 680-688.
[12] Amal Alsubaih, Alaaeldin Hafez, Khaled Alghathbar," Authorization as a Service in Cloud Environment" 2013 IEEE Third International Conference on Cloud and Green Computing, pp. 487-493.
[13] Nelson Mimura Gonzalez, Marco Antônio Torrez Rojas, Marcos Vinícius Maciel da Silva,Fernando Redígolo, Tereza Cristina Melo de Brito Carvalho,Charles Christian Miers, Mats Näslund,Abu Shohel Ahmed," A framework for authentication and authorization credentials in cloud computing", 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 509-516

[14] Ehtesham Zahoor, Olivier Perrin and Ahmed Bouchami "CATT: A Cloud Based Authorization Framework with Trust and Temporal Aspects", COLLABORATECOM 2014,October 22-25, Miami, United States.

