Access control delegation in the clouds

Pavani Gorantla

ACCESS CONTROL DELEGATION IN THE CLOUDS

by

PAVANI GORANTLA

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2014

Approved by

Dr. Dan Lin, Advisor

Dr. Wei Jiang

Dr. Sriram Chellappan

ABSTRACT

Current market trends need solutions/products to be developed at high speed. To meet those requirements sometimes it requires collaboration between the organizations. Modern workforce is increasingly distributed, mobile and virtual which will incur hurdles for communication and effective collaboration within organizations. One of the greatest benefits of cloud computing has to do with improvements to organizations communication and collaboration, both internally and externally. Because of the efficient services that are being offered by the cloud service providers today, many business organizations started taking advantage of cloud services. Specifically, Cloud computing enables a new form of service in that a service can be realized by components provided by different enterprises or entities in a collaborative manner. Participating parties are usually loosely connected and they are responsible for managing and protecting resources/data entrusted to them. Such scenario demands advanced and innovative mechanisms for better security and privacy protection of data shared among multiple participating parties.

In this thesis, we propose an access control delegation approach that achieves federated security services and preserves autonomy and privacy sharing preferences of involved parties. An important feature of our mechanism is that each party will not need to reveal its own sensitive information when making a global decision with other collaborators, which will encourage a wide range of collaboration and create more business opportunities.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who have helped me with this research. First, I would like to thank my advisor Dr. Dan Lin who gave me the opportunity to work on this research topic. Her valuable insights and suggestions have helped me to overcome many hurdles during this work. Secondly, I would like to thank Dr. Sriram Chellappan and Dr. Wei Jiang for being part of my thesis committee and taking time to review this work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The modern workforce is increasingly distributed, mobile and virtual [16]. Thus there will be many hurdles for communication and effective collaboration within organizations. One of the greatest benefits of cloud computing has to do with improvements to organizations communication and collaboration, both internally and externally. Thus by switching to the cloud, corporate resources can be virtualized, enabling individuals to access the documents they need regardless of location or device. Several cloud's web-based tools are developed to reduce communication barriers by helping people connect to the organizations cloud and get relevant and timely responses. For example, Event Industry Veteran had launched an EventCollab- cloud based collaboration software service (SaaS) [17]. It helps professionals to collaborate with stake holders within every project so that everyone involved in their project is on the same page. Moreover, cloud service providers also collaborate among themselves in order to provide better services to their customers. For example, Apple Inc. collaborates with amazon's AWS and Microsoft's Azure to host its iCloud services [18]. Oracle teams up with Amazon AWS to extend its services to customers [19]. Oracle collaborated with Microsoft for providing Microsoft Azure customers with oracle software services [20]. Cloud computing collaboration and communication suite of Sales force and Google Apps enables users of Sales force and Google Apps to collaborate more effectively using the cloud [21]. Hewlett-Packard (HP) collaborated with Sales force cloud service provider [22]. Sales force thus runs a dedicated instance of HP's coverage infrastructure on its cloud, providing a continuous service to HP's customers. As we can see from the above examples, cloud computing enables a new form of service in that a service can be realized by components provided by different enterprises or entities in a collaborative manner. Participating parties are usually loosely connected and they are responsible for managing and protecting resources/data entrusted to them. Such scenario demands advanced and innovative mechanisms for better security and privacy protection of data shared among multiple participating parties.

In this thesis, we propose an access control delegation approach that achieves federated security services and preserves autonomy and privacy sharing preferences of involved parties. Our proposed policy decomposition approach decomposes a global policy that needs to be enforced among participating collaborators. After the decomposition, the access control rights will be delegated to corresponding parties based on information available at each local party. Given a request to access certain information, the request will be evaluated locally at respective participating parties. Then, the local decisions will be assembled to make the final decision. In this way, each party will not need to reveal its own sensitive information when making a global decision with other collaborators.

We cast our solution in the context of the eXtensible Access Control Mark-up Language (XACML) [2] framework. XACML is a general purpose access control policy language which defines a request/response language and framework to enforce authorization decisions. We have chosen XACML because of its widespread adoption as a language of choice for enforcing access control in traditional and distributed environments [7]. In a typical XACML framework, there is a policy enforcement point (PEP) and a policy decision point (PDP). The PEP is responsible for issuing requests and enforcing the access control decisions. The PDP receives requests from the PEP and evaluates policies applicable to the requests and sends a decision back to the PEP. To support collaborative access control, we extend the XACML reference architecture by introducing multiple PDP's that communicate with a centralized PEP through a request dispatcher/decision coordinator. If the PDP's are at different hierarchical level, then that PDP will have child PDP's. A global policy is thus decomposed into local policies for each PDP according to availability/ sensitivity requirements of each party. Given a request, the central PEP modifies the request and dispatches it to corresponding PDPs, and then combines the decisions.

The other issue which we are focusing in this thesis is, generally even if a single policy in a global policy is modified or even if a single resource location has been modified, then the entire global policy will be re-evaluated which will incur lots of overhead. So, in this thesis we addressed these issues. We found a way to evaluate only those particular policies with modified resource locations or modified policies instead of evaluating whole global policy.

The rest of the thesis is organized as follows. Chapter 2 reviews the related work; Chapter 3 gives the details of our approach; Chapter 4 discusses the experimental results and chapter 5 gives the conclusion for our thesis

2. RELATED WORK

In this section, a review of XACML policies and significant work in access control delegation is presented.

2.1 INTRODUCTION TO XACML: (Extensible Access Control Markup Language)

XACML [2] is the OASIS standard language which is used to specify access control policies. These policies are expressed in XML form. It provides a common language to express security policies [2]. Here access control decisions are obtained by a request/ response sequence. The request contains details of subject (User who makes request to the resources), Action (An operation on Resource), Resource (Data, System component or Service) and environmental conditions (Set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action). So, requests finds out if the requesting user is allowed to perform a specific action on a particular resource under a given set of environmental conditions. The response will be a decision if the user can access the resource or not and obligations associated with the decision. The decision could be Permit, Deny, Indeterminate or Not Applicable).

XACML policies include three main components. They are Target, Rule set and Rule combing algorithms. Target defines a set of conditions in the policy that determine if the policies apply to a particular request. Rule set contains optional Target, a Condition and an Effect element.

Architecture of XACML engine is as follows.

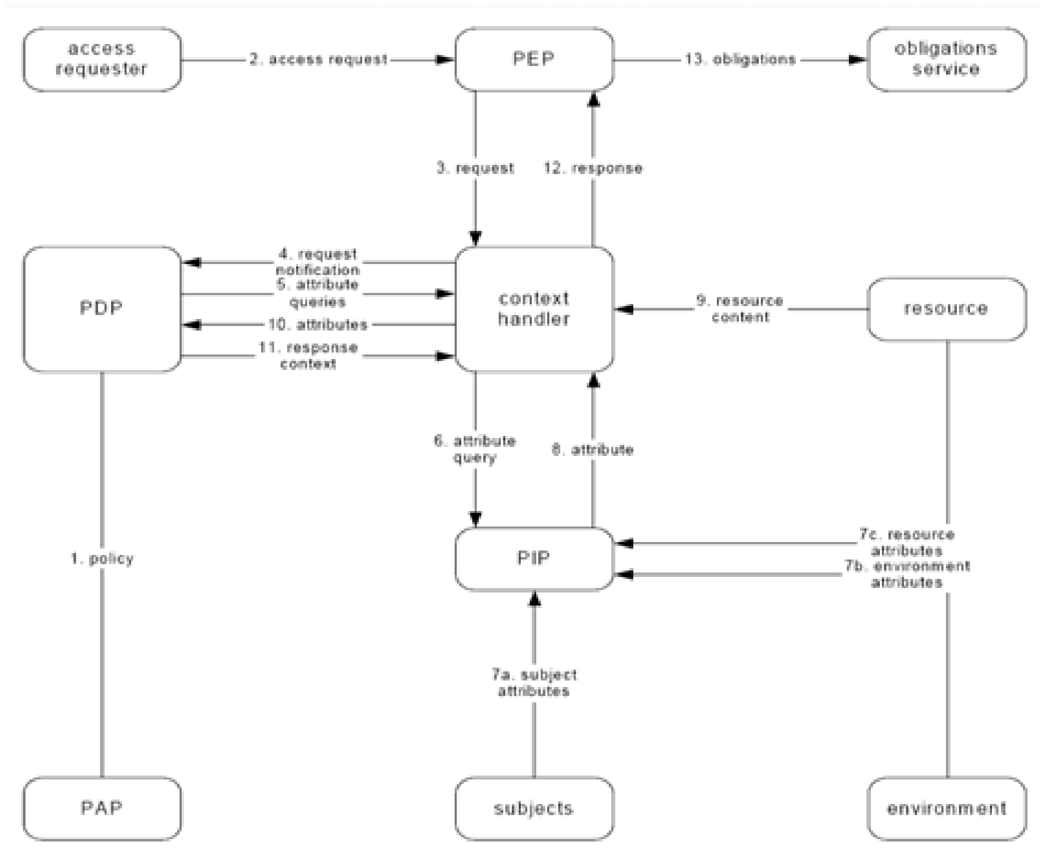


Figure. 2.1. An Architecture of XACML Engine

Let’s see what individual block does:

PEP (Policy enforcement point): It makes request/ response calls to the system.

PAP (Policy administration point): It creates security policies and stores them in repository.

Context Handler: Context Handler converts the requests in its native format to the XACML canonical form and to convert the Authorization decisions in the XACML canonical form back to the native format.

PDP (Policy Decision Point): Functionality of PDP is to receive and examine the request, retrieve the policies that are applicable, evaluate policies and send output to PEP.

PIP (Policy Information Point): PIP contains data required for policy evaluation

First policy is fed into PAP which stores the security policies. When access requester sends request to PEP, it sends the request to context handler. Context handler then notifies PDP about the request and retrieves attribute queries from PDP. Context handler then sends the attribute query to PIP. PIP will get all the required attributes and will send then to PDP. PDP will then evaluate the request and send the response to context handler in XACML form. Then the context handler will convert the context to native response and send the response to PEP. PEP will full-fill the obligations and sends the response.

2.2 ACCESS CONTROL DELEGATION IN CLOUD SYSTEM

Some of the access control delegation systems that have been proposed in cloud are as follows:

2.2.1. Policy Decomposition for Collaborative Access Control[10]. This policy decomposition was designed for the multi-party collaborative environment. In such environment decisions needs to be taken by different parties and then decisions from different organizations are grouped to obtain the final decision. In their system, they have a single central PEP (Policy Enforcement Point) which will take the global policy as an input, several PDP's (Policy Decision Point) which will be policies related to a particular organization, local repository for each PDP where the policy evaluation in a particular organization take place and a request dispatcher/decision coordinator. Request dispatcher/decision coordinator connects Central PEP and PDP's. Their system performs two main operations policy decomposition and request evaluation. In policy decomposition the global policies are divided into local policies based on decomposition constraints. These decomposed policies are sent to local policy repositories corresponding to their particular PDP's. Then the policies are evaluated in the corresponding local repositories and the output is then collaborated. But, they didn't take hierarchical level into consideration. The thesis we proposed is an extension to this project.

2.2.2. Automated Decomposition of Access Control Policies[9]. In dynamic distributed information systems the resources are distributed in multiple levels of hierarchy. This policy decomposition strategy was designed to address policies that need access to resources that are in different levels of hierarchy. Access control at higher level should be able to define who is allowed to use the resources. At lower levels, policy should be able to define if the user can access requested concrete resource or not. In this paper, they proposed a system which will automatically produce lower level policies from higher level policies. Lower level policies are then distributed to different concrete resources that use existing access control decision system. But, they didn't take autonomy of individual into consideration

2.2.3. Privacy Preserving Delegated Access in Public Clouds[5]. Enforcing fine grained access control on confidential data hosted in cloud incurs overhead to the data owner. So, in this approach they resolved that problem. In this approach they proposed a model which will delegate the fine-grained access to the cloud. Here, they enforced two layers of encryption. User implements coarse grained access control (inner layer) to encrypt the data and then fine grained access control (Outer Encryption layer) is performed on the encrypted data for controlling access to data. They proposed an algorithm for decomposition of policies and demonstrated improved performance. In order to achieve this policy refinement, described high level policy specifications, resource type hierarchies, and decomposition rules are required. Then they fed these into an inference engine to infer low level policies. But, they didn't take collaborative environments, hierarchical levels into consideration.

2.2.4. Ensuring Access Control in Cloud Provisioned Healthcare Systems[6].

Here they have analyzed the requirements of access control for health care multi-tenant cloud systems. They proposed a model to adapt task-role based access control. They considered privileges such as separation of duty, delegation of tasks, spatial and temporal access into consideration for giving access to users. But, they didn't take collaborative environments, hierarchical levels into consideration.

3. POLICY DECOMPOSITION

This policy decomposition is suitable for multiparty cloud collaborative environments. We have considered cloud as our platform since cloud has more resources and have global policies involving policies authorization from different organizations. The collaborating parties can be either a group of individual organizations or a single organization with several departments.

Architecture of our collaborating access control is based on the figure 3.1. Figure 3.1 also shows the information flow. The basic idea is to decompose a global policy in such a way that each participating party does not need to have any sensitive information belonging to other parties to make an access control decision, and to combine decisions made by each participating party to obtain the decision for the global policy.

In our system, there is a central policy enforcement point (PEP) and many parent policy decision points (PDP) which will in turn collaborate decisions from many local PDP's. The central PEP and PDP's are connected by request dispatcher/decision coordinators (RDDC). The PEP, RDDC, policy decomposition module, global policy repository reside at one party called coordinator; each PDP and associated local policy repository reside at each collaborating party. The system implements two key functions: policy decomposition and request evaluation.

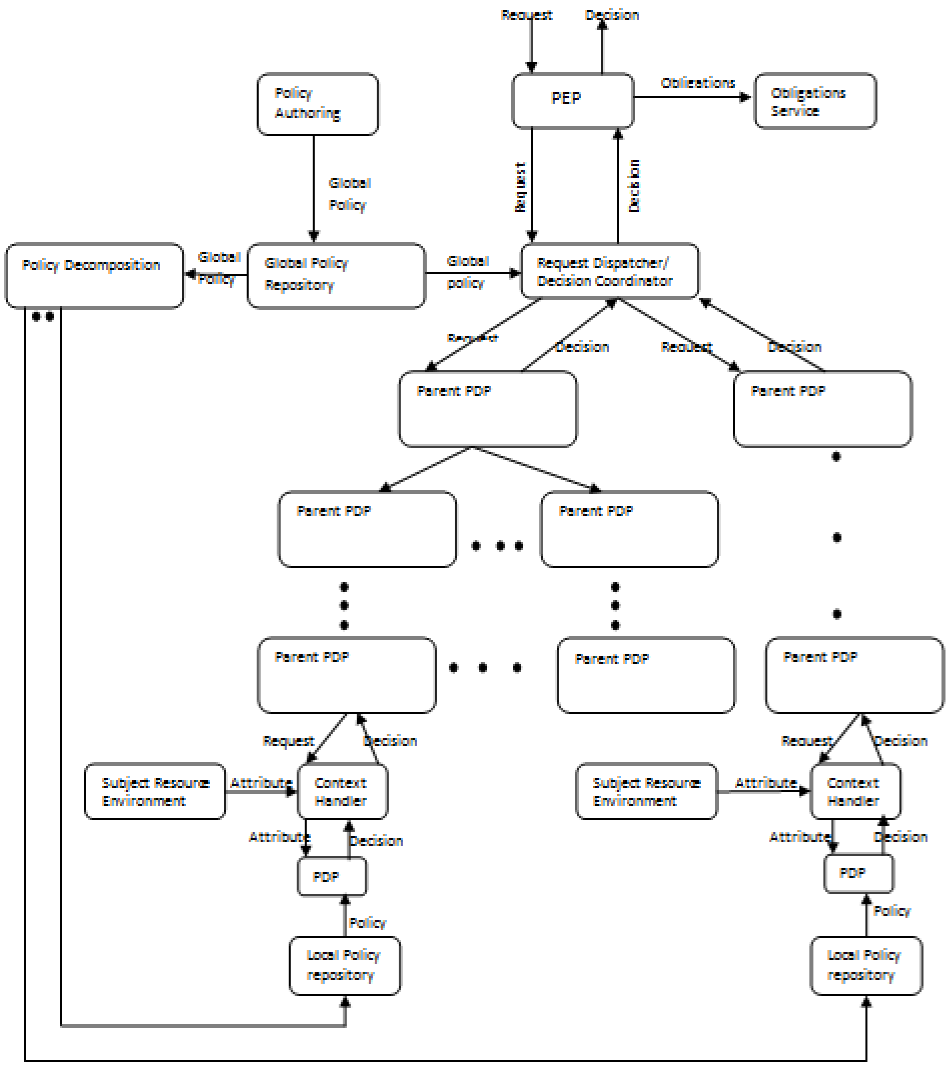


Figure. 3.1. Data Flow Diagram

The policy decomposition function takes a global policy as input. The global policy is decomposed into local policies and then sent to the local policy repositories of corresponding PDPs. This function is performed by the trusted coordinator. After the decomposition, the global policy is encrypted and stored in a secure store. That means that the global policy will no longer be used for the subsequent request evaluation.

Instead, only the non sensitive information of each global policy is kept as plain text in a policy table maintained by the coordinator. These rules are decomposed into local policies and sent to corresponding local/ parent PDP's. The parent PDP's will in turn send's the policies to corresponding local/ Parent PDP's. This will continue till the final local PDP is reached. In short, the coordinator and parent PDP's are responsible for coordination and does not maintain any sensitive information; sensitive information is stored at each local PDP.

The request issued contains subject attributes, resource attributes, action attributes and environmental attributes. Targets mentioned will determine if the policy can be applied to the given request. Resource attributes refer to the service. Action attributes determines the action user wants to perform on the requested service. Environmental condition refers to the attributes that helps in making authorization decision and whose conditions are independent of a subject, resource and action. If the target matches, then the request dispatcher/ decision coordinator will send requests to particular parent PDP which in turn will send requests to underlying PDP. This will continue till we find a final local PDP. Each local PDP is associated with local repository. When final PDP is reached, the requests will be evaluated in the PDP and the decisions obtained from all the local PDP's will be grouped in respective parent PDP's and the final decision will thus be obtained by grouping the decisions obtained from all local/parent PDP's in PEP.

Policy decomposition function will take a global policy as an input. In our system, we assumed that the global policies are arranged in DNF (Disjunctive normal form) [29]. We will decompose the global policy and send them to local PDP's and save the details of the decomposed policies in coordinator or parent PDP's. The details include which policy belongs to which PDP, how the decisions are grouped, policy id's etc.

The request issued contains subject attributes, resource attributes, action attributes and environmental attributes. Targets mentioned will determine if the policy can be applied to the given request. Resource attributes refer to the service. Action attributes determines the action user wants to perform on the requested service. Environmental condition refers to the attributes that helps in making authorization decision and whose conditions are independent of a subject, resource and action.

3.1 AN ILLUSTRATIVE EXAMPLE

To illustrate an example we are considering a set of collaborated organizations. Let us assume the hierarchical structure of the organization is as in figure 3.2.

So, let us say that the global policy is "If an employee belongs to organization C and working in training department in organization B with accounts payable less than 10,000 and funding more than 10,000,000 or an employee from organization A and working in training department in organization B with funding more than 10,000,000 can buy advanced equipments." This policy contains two rules. The rule P.r1 states that the employee who is working on project "access control" and who actually belongs to organization C and performing a collaborative operation in training department of organization B and having funding to buy an equipment more than 10,000,000 with accounts payable by him less than 10,000 can buy advanced equipments. The rule P.r2 states that the employee who is working on project "access control" and who actually belongs to organization A and performing a collaborative operation in training department of organization B and having funding to buy an equipment more than 10,000,000 can buy advanced equipments.

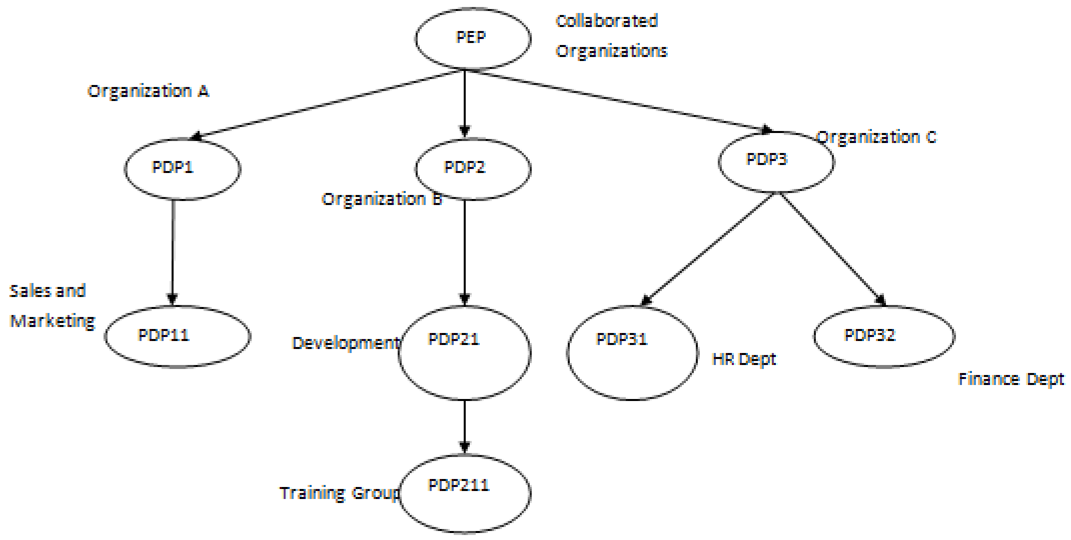
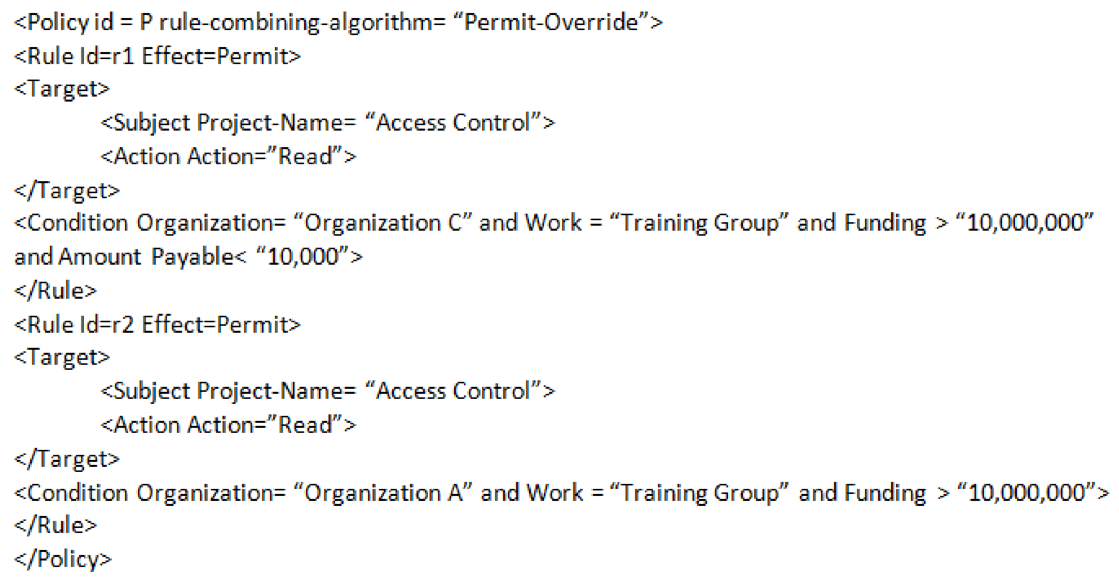


Figure. 3.2. An Example of Hierarchical Organization Structure

So, XACML global policy will be as shown in figure 3.3. We further assume that "Project Name" and "Action" are public information and known by any organization, while information about "accounts payable" and "Funding" is stored in the finance department in organization C and information about "organization A" is stored in organization A and information about "organization C" is stored in organization C and information about training department is stored in organization B.

According to the available information at each department, we decompose the policy P into P1, P2, P3 and P4 with permit effect as follows, where policy P1 contains information about organization C, P2 contains information about organization B, P3 contain about organization A and P4 only contains financial department information.



```
<Policy id = P rule-combining-algorithm= "Permit-Override">
<Rule Id=r1 Effect=Permit>
<Target>
            <Subject Project-Name= "Access Control">
            <Action Action="Read">
</Target>
<Condition Organization= "Organization C" and Work = "Training Group" and Funding > "10,000,000"
and Amount Payable< "10,000">
</Rule>
<Rule Id=r2 Effect=Permit>
<Target>
            <Subject Project-Name= "Access Control">
            <Action Action="Read">
</Target>
<Condition Organization= "Organization A" and Work = "Training Group" and Funding > "10,000,000">
</Rule>
</Policy>
```

Figure. 3.3 An Example of an XACML Global Policy

P1 (Organization C): Any employee of project "Access Control" working in organization C and having funding> 10,000,000 and amount payable< 10,000 can buy advanced equipment's.

P2 (Organization B): Any employee of project "Access Control" working in training group in developer department in organization B.

P3 (Organization A): Any employee of project “Access Control” working in organization A can buy advanced equipment’s.

P4 (Organization C): Any employee can buy advanced equipment’s for the project “Access Control” with funding more than 10,000,000 dollars.

In the above example both P1 and P4 are checking for condition funding > 10,000,000. To avoid such redundant evaluation and improve the efficiency, our system will simplify policy P1 as follows.

P1' (Organization C): Any employee of project “Access Control” working in organization C and amount payable< 10,000 can buy advanced equipment’s.

P4 (Financial Department): Any employee can buy advanced equipment’s for the project “Access Control” with funding more than 10,000,000 dollars.

We will then group the policies those belong to same department and send those clustered policies to particular PDP. Here, policies P1' and P4 belong to same parent, PDP3. We will cluster these policies together and will send them to PDP3. We will send the policy P2 to PDP2 and policy P3 to PDP1.

In PDP3, we will further decompose the cluster {P1', P4} based on PDP’s. Here P1' belongs to PDP3 and P4 belongs to PDP32. So, we can decompose policy P1 ' as follows:

P5(Organization C): Any employee of project “Access Control” working in organization C can buy advanced equipment’s.

P6(Financial department): Any employee of project “Access Control” and amount payable< 10,000 can buy advanced equipment’s.

Policy P4 belongs to PDP32 and cannot be further decomposed. The Policy P5 and P6 cannot be further decomposed. The PDP3 contains details only about P5. PDP3 doesn’t contain the information about policies P6 and P4 belong to PDP32. So, we will cluster the policies {P6, P4} and send the policies to PDP32. So, the local policy will be stored in PDP32. The policy P3 cannot be further decomposed and PDP1 contains information about the P3. So, we will store P3 in local repository of PDP1. Policy P2 cannot be further decomposed. But PDP2 doesn’t contain details about the P2. So, we will check to which PDP P2 belongs to in PDP2. Since the policy belongs to PDP21, we will send the policy to PDP21. PDP21 doesn’t contain the details about P2. Since the policy belongs to PDP211, we will send the policy to PDP211. It contains the details of policy P2. So, we will store the policy to PDP211.

We will maintain the details of initial policy decomposition in coordinator and details of further decompositions in parent PDP’s. We will clearly see how we will perform the policy decomposition and how we will evaluate the request in next section.

3.2 HIERARCHICAL DECOMPOSITION:

3.2.1. Decomposition Strategy. This work presented in this thesis is an extension of project “Policy Decomposition for Collaborative Access Control” [10]. There they have proposed a novel approach for global policy decomposition among collaborative parties.

We are using the same decomposition strategy for policy decomposition, but in this paper we are considering the hierarchical relationships among PDP's where each PDP reports the decision to its parent PDP.

Algorithm for Hierarchical Policy Decomposition (P):

Input: P is a global policy.

1) For each rule ri in P, create a compound Boolean expression for ri.
2) Label each atomic Boolean expression.
3) Decompose the policies and construct local policies.
4) After decomposition cluster the policies that belong to same PDP
5) Distribute the clustered policies to the destination PDP
6) Construct the final effect combination table for each rule at PEP
7) Perform steps 3 to 5 till every local policy in a cluster reaches the final Local PDP and construct effect combination table at each PDP.

Let us see the working of the algorithm with an example. Consider the policy P defined in figure 3.3

Step1: In the policy P, we have two rules r1 and r2. They can be represented as follows. Where {}T represents targets.

P.r1 = {(Project Name= “Access Control”) ^ (Action= “Buy”)}T ^ (Organization= “Organization C”) ^ (Work= “Training Group Organization B”) ^ (Funding>10,000,000) ^ (Amount Payable< 10,000)

P.r2 = {(Project Name= “Access Control”) ^ (Action= “Buy”)}T ^ (Organization= “Organization A”) ^ (Work= “Training Group Organization B”) ^ (Funding>10,000,000)

Step2: Labeling each atomic Boolean expression

So based on the table 3.1, rules P.r1 and P.r2 can be represented as follows:

- P.r1: B1(L1) ^ B2 (L2) ^ B3(L1) ^ B4(L1) ^ {B6(Ls) ^ B7(Ls)}T
- P.r2: B5(L3) ^ B2 (L1) ^ B4(L1) ^ {B6(Ls) ^ B7(Ls)}T

Step 3: So, we can decompose the policies as

- P1: B1(L1) ^ B3(L1) ^ B4(L1) ^ {B6(Ls) ^ B7(Ls)}T
- P2: B2 (L2) ^ {B6(Ls) ^ B7(Ls)}T
- P3: B5(L3) ^ {B6(Ls) ^ B7(Ls)}T
- P4: B4(L1) ^ {B6(Ls) ^ B7(Ls)}T

The policy B4 (L1) is executed both in policy P1 and in policy P4. So, we will decompose the repeated policies as follows:

- P1`: B1(L1) ^ B3(L1) ^ {B6(Ls) ^ B7(Ls)}T
- P2: B2 (L2) ^ {B6(Ls) ^ B7(Ls)}T

- P3: B5(L3) ^ {B6(Ls) ^ B7(Ls)}T
- P4: B4(L1) ^ {B6(Ls) ^ B7(Ls)}T

Table. 3.1. Atomic Boolean Expressions and Labeling

ID	Unique atomic Boolean Expression	Label
B1	Organization= “Organization C”	L1
B2	Work= “Training Group”	L2
B3	Amount payable < 10,000	L1
B4	Funding > 10,000,000	L1
B5	Organization = “Organization A”	L3
B6	Project Name= “Access Control”	Ls
B7	Action = “Buy”	Ls

Where L1, L2, L3 indicates they are from different departments

Step 4: Here policies P1`, p4 belong to same PDP (PDP3), so group the policies together and send them to PDP3 as a cluster. Send P2 to PDP2 and P3 to PDP1. So, the three clusters available here are shown in Table 3.2

Table. 3.2. Clustered policies at PEP

Cluster	PDP	Resource Found
C1:{P1`,P4}	PDP3	False
C2:{P2}	PDP2	False
C3:{P3}	PDP1	True

Step 5: Distribute these clustered policies to respective PDP’s

Step 6: Effect combination table at PEP can been seen in Table 3.3

Step 7: Perform steps 3 to 5 till every local policy in a cluster reaches the final Local PDP and construct effect combination table at each PDP.

Table. 3.3. Effect combination table at PEP

RID	F
P.r1	e(P1`) ^ e(P2) ^ e(P4)
P.r2	e(P3) ^ e(P2) ^ e(P4)

Based on the PDP values we further decompose these policies

For C1 at PDP3,

P1` = {(Project = “Access Control”) ∧ (Action= “Buy”)}T ∧ (Organization= “Organization C”) ∧ (Amount Payable< 10,000)

P5= P1`.c1 = B1 ∧ {B6 ∧ B7}T

P6= P1`.c2 = B3 ∧ {B6 ∧ B7}T

P4 = {(Project = “Access Control”) ∧ (Action= “Buy”)}T ∧ (Funding>10,000,000)

Where P4 cannot be further decomposed,

P5 will be sent to PDP3, which is the final PDP for P5. So, the local policy P5 is stored in PDP3. P6 and P4 belong to same PDP (PDP32); we will cluster these policies and will send them to PDP32. PDP32 is the final PDP for policies P6 and P4. So, they will be stored in PDP32.

Table. 3.4. Clustered policies at PDP3

ID	PDP	Resource Found
P5	PDP3	True
P6	PDP32	True
P4	PDP32	True

Effect combination table for PDP3 is as can be seen in Table 3.5

Table. 3.5. Effect combination table for PDP3

Policy ID	Effect
P1`	e(P5) ^ e(P6)
P4	e(P4)

For cluster C2, in PDP2,

P2= {(Project = “Access Control”) ∧ (Action= “Buy”)}T ∧ (Work= “Training Group Organization B”)

P2 cannot be decomposed further, but PDP21 doesn’t contain the details regarding P2. So we will send the cluster C2 that contains P2 to next sub department that contains details of Policy P2.

So, the clustered policies at PDP2 can be seen in Table 3.6

Table. 3.6. Clustered policies at PDP2.

ID	PDP	Resource Found
P2	PDP21	False

So, the effect combination table for PDP2 is shown in Table 3.7

For cluster C2, in PDP21, P2 cannot be decomposed further, but PDP21 doesn't contain the details regarding P2. So we will send the cluster C2 that contains P2 to next sub department that contains details of P2.

Table. 3.7. Effect combination table for PDP2

Policy Id	Effect
P2	e(P2)

So, the clustered policies at PDP21 can be seen in Table 3.8

So, the effect combination table for PDP2 is shown in Table 3.9

Table. 3.8. Clustered policies at PDP21

ID	PDP	Resource Found
P2	PDP211	True

Here, the policy P2 will be evaluated. For cluster C3, in PDP1, PDP1 is the final resource. Since, all the final local PDP's are reached, we will stop the decomposition.

Table. 3.9 Effect Combination Table at PDP21

Policy Id	Effect
P2	e(P2)

After performing policy decomposition our main concern is to cluster the policies that belong to same PDP which will reduce the number of calls to that particular PDP. Let us assume there are different policies p1, p2, p3, p4, p5, p6, p7, p8, p9, p10. {p1, p2, p3} ϵ D1. {p4, p5, p6} ϵ D2 and {p7, p8, p9, p10 } ϵ D3. If a policy set in the global policy is as follows (p1 ∧ p4 ∧ p7 ∧ p2 ∧ p5 ∧ p8 ∧ p3 ∧ p6 ∧ p9 ∧ p10). The traditional XACML policy will execute one policy at a time. In our approach, we will group the policies from same department together. That is, we will send (p1∧p2∧p3) to D1,

($p4 \wedge p5 \wedge p6$) to D2 and ($p7 \wedge p8 \wedge p9 \wedge p10$) to D3 respectively. This will reduce the time number of calls to Local PDP.

3.2.2. Request Evaluation. Now, let us see how request evaluation takes place. Let us assume that Bob working in organization C and is working in a training group department of organization B as a part of "Access Control" project and has a funding of 50,000,000, has the accounts repayable as 5,000 and he wants to buy an equipment. Corresponding request <Bob, Project-Name= "Access Control", Action= "Buy"> is received by the coordinator, the coordinator will check if the targets of the request matches with the targets in the global policy. If the target matches, then it sends the request to the organization A, Organization B and Organization C. The organization C will evaluate the request to permit if the employee belongs to organization C and will send the policies related to financial department to next level. The policies related to financial department are evaluated here and the output is sent to organization C, there the decisions will be combined. Similarly, it will transfer the details regarding employee to training group and will evaluate if Bob belongs to training department or not. The request will be evaluated in training group department. Finally, all the outputs are grouped in final PEP. Similarly the policies will be sent to other departments. Since Bob satisfies all the conditions specified in P.r1, P.r1 will return permit decision. Since, one rule returns permit and since the rule combining algorithm is "Permit- Override", the above request will return Permit decision.

A straightforward approach to evaluate a request consists of three basic steps: (i) for each rule applicable to the request, evaluate its local policies; (ii) In Parent PDP combine the decisions based on effect combination table; and (iii) apply the policy combining algorithm at PEP to obtain the final decision of the request.

Different policies may share the same local policies and hence some policies may be repeatedly evaluated. So, in our approach we will not reevaluate the policies that are repeated. Consider the permit-override combining algorithm as an example. If a rule with the permit effect is evaluated true, we do not need to check other rules, i.e., we do not need to check corresponding local policies.

Two main data structures are used in our method. IRE is an intermediate result table which stores the effects of local policies on a given request. RS is a response time table which keeps a record of evaluation time of each rule and each local policy.

We will store the output of that particular policy in an IRE table. We will get the output from IRE table if the policy is repeated again. If the policy combining algorithm is permit- override, if one policy returns permit, we will stop the execution of remaining policies and will send the decision to the user. Similarly, based on the policy combining algorithms available, we will evaluate the request till one policy set returns a permit decision

So, the request evaluation algorithm will be as follows:

```
Algorithm for Request_evaluation(q,G)

    Input: q is a request regarding policy P.

    1)  For each rule applicable to the request, evaluate its local policies
```

2) Combine the effects of local policies based on effect combining tables.

3) Store the decisions in Intermediate result table (IRE) and response time in Response time table (RS)

4) If the policies are repeated, then get the output from IRE table

The other areas which we focused on are:

3.3 UPDATING OF POLICIES

One of the drawbacks of the existing work is that if there is a small change in already available global policy or if the resource location has been moved, then the whole policy will be recompiled, which consumes lot of time. So, here we addressed this problem. There are two cases in this.

3.3.1. When the Global Policy is Updated. Here, first we will check if the global policies are updated by computing “Levenshtein Distance” [28]. If the policies are updated, then we will check how many policies have been updated

The below mentioned algorithm explains how does the algorithm work.

Algorithm for Update of global policies:

1) Then, we will check if there is a change in global policy by computing “Levenshtein Distance” [28].

2) If there is a change in global policy, then we will execute only those requests corresponding to modified PDP’s.

3) For others, we will store the results obtained from the IRE

Let us see how the algorithm works. For simplicity let us assume the policy as follows. Let us assume the values of a=3, b=4, d=6 and e=4 and the initial global policy is $G=((a<3) \wedge (b<6) \wedge (d>5) \wedge (b>3)) \vee ((b>2) \wedge (a<3) \wedge (d<7) \wedge (e>3))$. So, the output list will be <ArrayList<ArrayList<Boolean>>>, which will be as follows [[false,true,true,true], [true,true,true,true]]

If the policy has been modified, as follows $G = ((a<5) \wedge (b<6) \wedge (d>5) \wedge (b>3)) \vee ((b>2) \wedge (a<3) \wedge (d<7) \wedge (e>3))$

Here a single condition is changed, that is, the condition a<3 is modified to a<5. Generally, if the policies are modified, then the entire policy will be reevaluated. So, in this approach by using leveinshtein distance to find which policies have been modified. We will only compute that one policy which has been modified. So, we will compute the output for that single policy and will update the value in already existing output list. So, we will calculate the value of a<5 which is true, and update that particular policy's Boolean value. So, the new output list will be [[true,true,true,true], [true,true,true,true]]. From there we will calculate the final output.

3.3.2. When the Resources Locations are Updated. There might be a situation where we need to move the resource from one department to another department. In such cases, instead of re-evaluating whole policy we can just re-evaluate the part of policies whose resource values have been changed.

Algorithm for Update of resources:

1) We will check if there is a change in old resources and newly allocated resources. If there is a change, then we will build a new hash table in which we will save the details like to which PDP the new resources now belong to

2) Then we will execute only those policies whose resource location is modified.

3) For other policies we will retrieve the information from the values stored in IRE table and compute the results

Let us see how the algorithm works. Let us say there are resources (r1, r2 and r3 in Organization1) and (r4, r5 and r6 in Organization2). The resource, r3 has been moved from organization 1 to organization2. So, we will find which resources have been modified. Here by comparing old and new resources and we will see that the resource r3 is added to Organization2.

Let us say the policy is as follows $(p1 \wedge p2 \wedge p3 \wedge p4) \vee (p3 \wedge p5 \vee p6)$. Here, the policy let us assume that the p1 operates on r1, p2 on r2, p3 on r3 , p4 on r4,p5 on r5 and p6 on r6. Let us say that the output list previously is as follows [[true,false,true,true], [true,true,true]]. Now since, the resource r3 has been moved we need to evaluate the policies p3. And find the output and modify the output for those policies and compute the output again.

4. PERFORMANCE STUDY

4.1 GENERATION OF DATASET

The datasets that we are generating for conducting the tests are global policies and requests.

Global policies contain set of policy sets, where each policy contains set of policies in it. Since, the policies will be in the form of Boolean expressions, we randomly generated some conditions. We generated the data in the form of DNF. Here we generated the policies for three levels of hierarchies. We are generating the policies for 20 different departments. In level one hierarchy, we will assume that there are 20 different departments. In second level we assume that each department has 10 sub departments in it. In third level of hierarchy we assume that each department again has 10 sub departments under them. We even assume that with increase in a level of hierarchy, retrieving the decision will get will get delayed by 1 millisecond. That is, if the policies are at level 1, the decision retrieval will take 1 millisecond; for level 2 it is 2 milliseconds and for level 3 it is 3 milliseconds. For conducting the tests we took 20 different global policies and evaluated the performance by taking the average of the 20 different policies. We assume that the global policy contains 10 policy sets and each policy set contains 10 policies.

We generated different request values randomly.

4.2 EXPERIMENTAL RESULTS

Here are some experiments we conducted to test the efficiency of the request evaluation structure defined.

4.2.1. Performance Measure at Different Hierarchical Levels. The purpose of this test is to check how the request evaluation time and policy decomposition time gets affected when evaluating the policies which are at different levels of hierarchy.

Here we are generating global policies involving policies only from 10 departments. Here, we are varying the hierarchy levels and we are even checking the performance, if the global policies are fully clustered, fully distributed and semi clustered.

a) Request evaluation time:

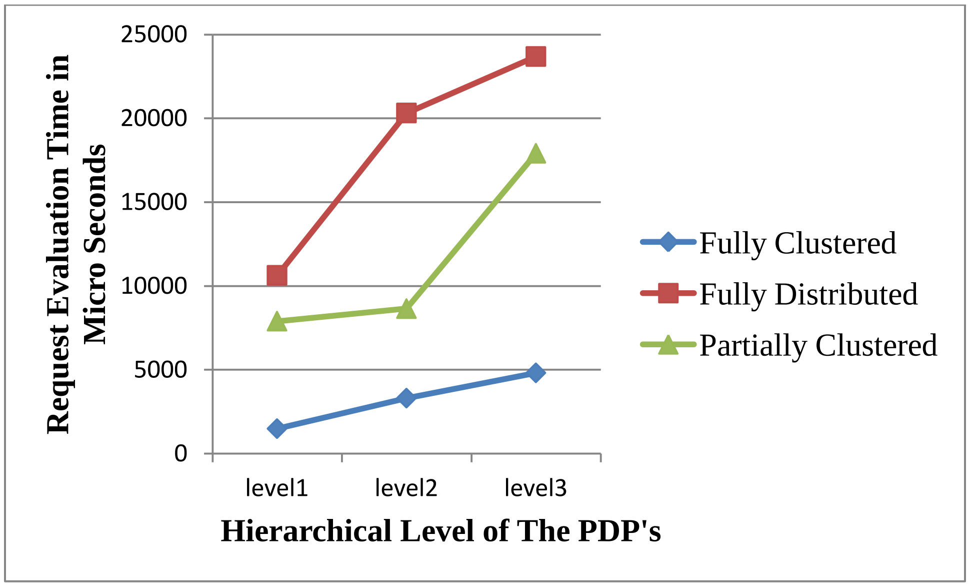


Figure. 4.1. Effect of Request Evaluation Time for different Hierarchy Levels

Figure 4.1 represents the results of request evaluation time for global policies with different hierarchy levels and with different clustered levels. We can observe here that with increase in hierarchical levels, the request evaluation time increase. We can even observe that the request evaluation time will be less if the policies are fully clustered and will be high if the policies are fully distributed.

This confirms that if policies are fully clustered then the evaluation time will be less. This is because, if the policies are fully clustered, then for evaluating these policies we will send all these policies to local PDP at once. So the number of calls to the PDP will be less. So, the evaluation time will be less. On other hand, if the policies are fully distributed then the evaluation time will be more. This even confirms that with increase in hierarchical levels, the request evaluation time increases. This is because, with the increase in level of hierarchy, the time taken to reach local PDP is high.

b) Policy decomposition time:

Here we are generating global policies involving policies only from 10 departments. Here, we are varying the hierarchy levels and we are even checking the performance, if the global policies are fully clustered, fully distributed and semi clustered.

Figure 4.2 represents the results of policy decomposition time for global policies with different hierarchy levels and with different clustered levels. We can observe here that increase in hierarchical levels doesn’t affect the policy decomposition much. But, clustering policies affect the policy decomposition time. That is, if the policies are fully clustered, then the policy decomposition time is high. But, if the policies are fully distributed, then the policy decomposition time will be less.

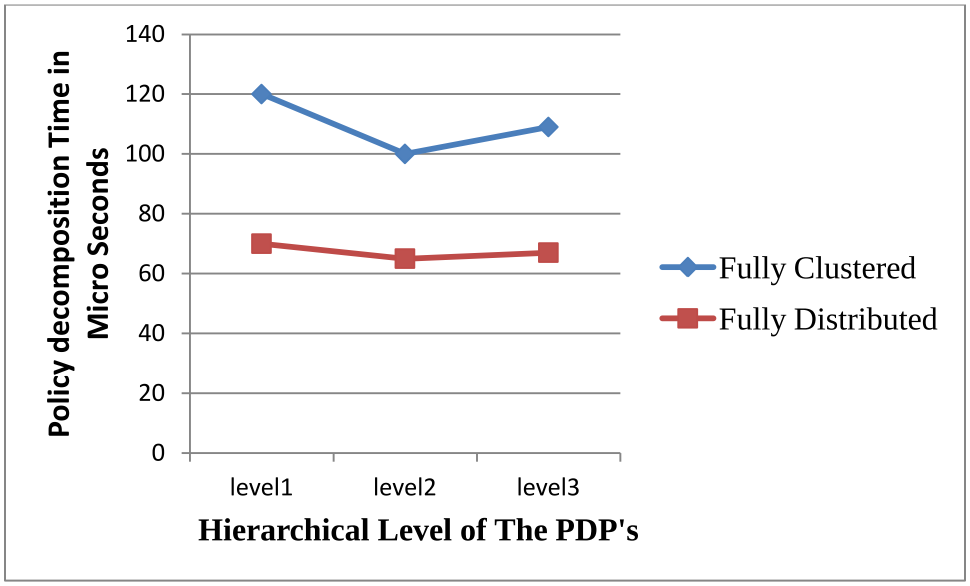


Figure. 4.2. Effect of Policy Decomposition Time for different Hierarchy Levels

This confirms that if policies are fully clustered then the policy decomposition time will be more. This is because if the number of policies to be clustered is more, it will take time to cluster the policies together. So, the policy decomposition time will be high. On the other hand, if the number of policies to be clustered is less, it will take less time for performing policy decomposition.

4.2.2. Performance Measure with Increase in Number of Departments. The purpose of this test is to check how the request evaluation time and policy decomposition time will get effected with increase in number of departments.

Here we are varying number of departments from 2 to 20. We are assuming that all the policies are from hierarchical level 2 and we are even checking the performance, if the global policies are fully clustered, fully distributed and semi clustered.

a) Request Evaluation time:

Here we are also checking the execution time if the execution is performed without any policy decomposition

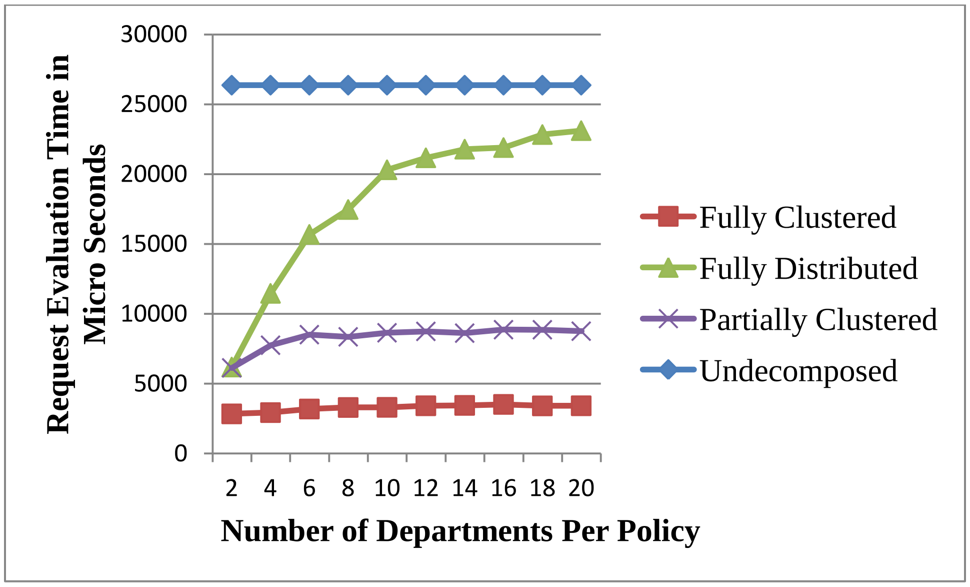


Figure. 4.3. Effect of Request Evaluation Time with Increase in Number of Departments

Figure 4.3 represents the results of request evaluation time for global policies with increase in number of departments and with different clustered levels. We can observe that with increase in number of departments, the request evaluation time increase. We can even observe that the request evaluation time will be less if the policies are fully clustered and will be high if the policies are fully distributed. We can even observe that the time taken to evaluate un-decomposed policies is more compared to time taken to evaluate policies after performing policy decomposition.

This confirms that if policies are fully clustered then the evaluation time will be less and if the policies are fully distributed than the evaluation time will be more. This confirms that with increase in number of departments, the request evaluation time increases. This is because increase in number of departments implies more PDP's, which implies more local PDP's to evaluate the request. This even confirms that policy decomposition will improve the performance of the request evaluation.

b) Policy decomposition time:

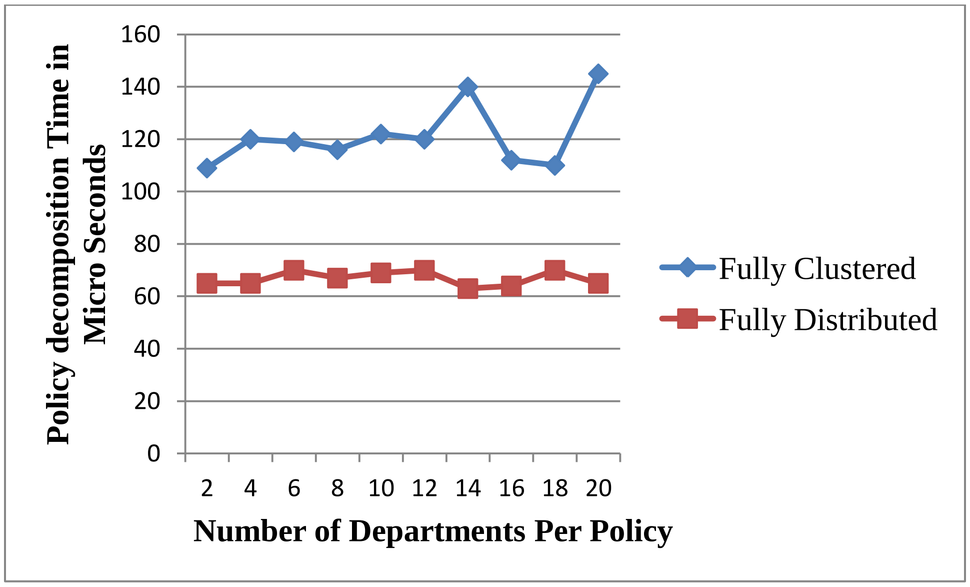


Figure. 4.4. Effect of Policy Decomposition Time with increase in number of departments

Figure 4.4 represents the results of policy decomposition time of global policies with variation in number of departments and with different clustered levels. We can observe here that increase in number of departments doesn't affect the policy decomposition much. But, clustering policies affect the policy decomposition time. That is, if the policies are fully clustered, then the policy decomposition time is high. But, if the policies are fully distributed, then the policy decomposition time will be less.

This confirms that if policies are fully clustered then the policy decomposition time will be more and if the policies are fully distributed that the policy decomposition time will be less.

4.2.3. Performance Measure with Variation of Number of Policies from Same Department. The purpose of this test is to check how the policy decomposition time will get effected with variation of number of policies from same department

Here we are varying the number of departments from which you retrieve the policies in policy set. We will vary the number of policies from same department from 1 to 5 and will measure the performance. We are keeping the default hierarchical level as level 2.

Figure 4.5 represents the results of policy decomposition time by varying the number of departments from same department from 1 to 5. We can observe here that with increase in number of policies from same department's decreases the request evaluation time.

So, from here we can confirm that if the policy set has all the policies, from same department then the request evaluation time will be less, which even satisfies the fully clustered condition. We can even observe that if the policy set has all policies from different department, then the request evaluation time will be more, which will satisfy the full distributed condition.

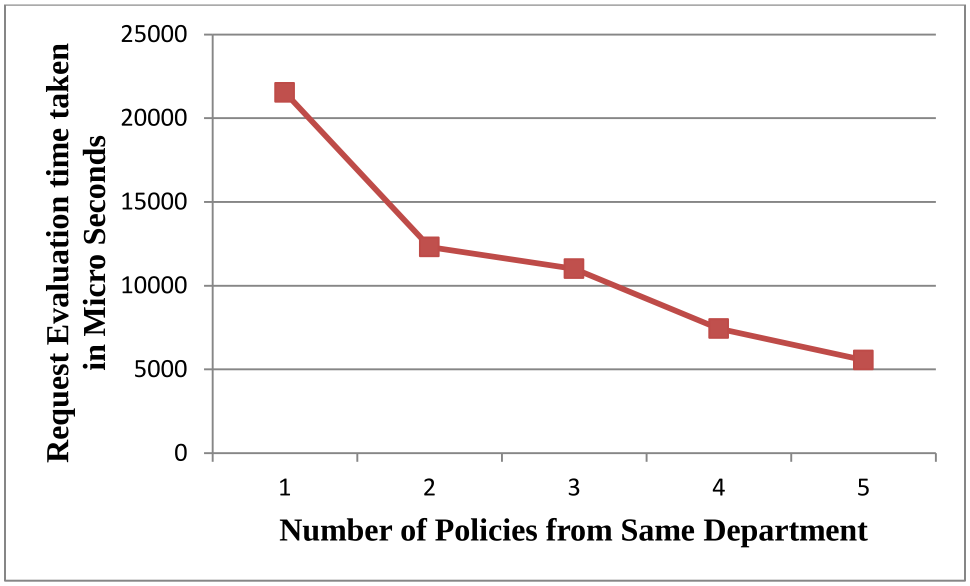


Figure. 4.5 Effect of Request Evaluation Time with increase in number of policies from same department

4.2.4. Performance Measure for Updating the Resources. The purpose of this test is to check how the request evaluation time will get effected with variation of number of resources to be updated

Here we are varying the number of resources to be updated from 10 to 100 and will measure the performance. We are keeping the default hierarchical level as level 2, and the number of departments as 10.

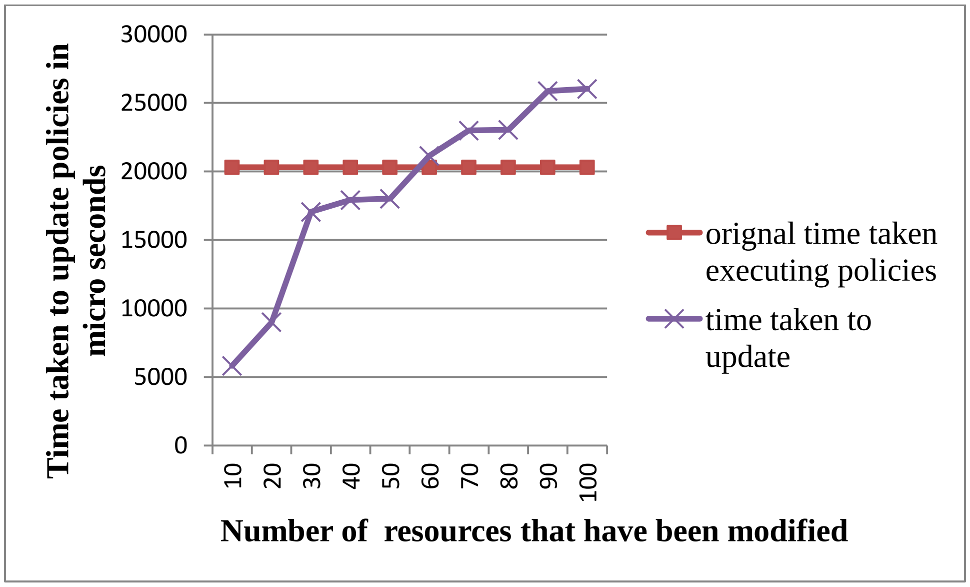


Figure. 4.6. Effect of Request Evaluation Time with increase in number of resources that are modified

Figure 4.6 represents the results of request evaluation time by varying the number of resources modified from 10 to 100. We can observe here that with increase in number of modified resources, the request evaluation time increases.

So, this confirms this approach is efficient if the number of resources modified is less than 60% of all the resources being used in the global policy. So, evaluating global policies when resources locations are modified without reevaluating entire policy increases the performance of the system if the number of resources being modified is less than 60%.

4.2.5. Performance Measure for Updating the Global Policy. The purpose of this test is to check how the request evaluation time will get effected with variation of percentage of global policy being updated

Here we are varying the percentage of global policy being updated from 10 to 100 and will measure the performance. We are keeping the default hierarchical level as level 2, and the number of departments as 10.

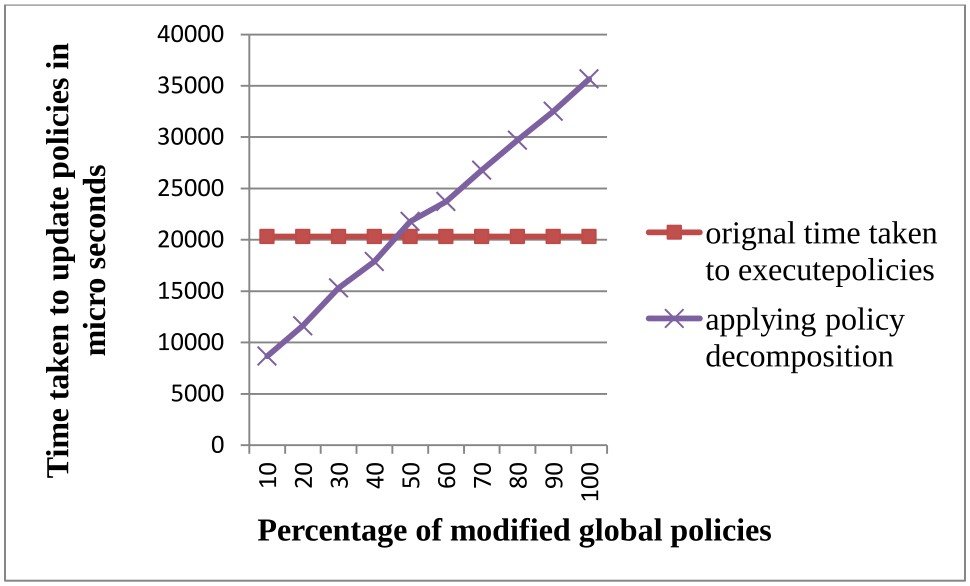


Figure. 4.7. Effect of Request Evaluation Time with increase in number of policies being updated

Figure 4.7 represents the results of request evaluation time by varying the percentage of global policy being modified from 10 to 100. We can observe here that with increase in percentage of modified global policy, the request evaluation time increases.

So, this confirms that this approach is efficient if the percentage of global policies modified is less than 50% of all policies available in global policies, then the request evaluation time for evaluating the policies will be less compared to reevaluating the whole policy. So, evaluating the global policies if the number of policies that are modified are less than 50% without reevaluating entire policy increases the performance of the system.

5. CONCLUSION

In this paper, we have proposed an access control model for collaborative access control in cloud environment. Our architecture is developed based on the XACML framework which allows our technique to be easily integrated into existing systems. The main idea is to properly decompose a global policy and distribute it to each collaborating party in the different hierarchical level. The decomposition ensures the autonomy and confidentiality of each involved party and guarantees the consistency of the decisions. Also, we proposed an algorithm to update the resources without reevaluating whole policy and proposed an algorithm to update the global policy without reevaluating whole policy.

REFERENCES

[1] Sun’s XACML open source implementation http://sunxacml.sourceforge.net

[2] Extensible access control markup language (XACML) version 2.0. OASIS Standard, 2005.

[3] Hierarchical resource profile of XACML v2.0 OASIS Standard, 1 February 2005.

[4] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah. First experiences using XACML for access control in distributed systems. In Proc. of ACM workshop on XML security, pages 25–37, 2003.

[5] M. Nabeel, E.West Lafayette Bertino. Privacy Preserving Delegated Access Control in Public Clouds. IEEE Transactions on Knowledge and Data Engineering PP(99): 1, 2013.

[6] H.A.J. Narayanan, M.H. Giine. Ensuring access control in cloud provisioned healthcare systems. IEEE Consumer Communications and Networking Conference (CCNC):247 – 251, 2011.

[7] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah. First experiences using XACML for access control in distributed systems. In Proc. of ACM workshop on XML.

[8] M. Nabeel and E. Bertino, “Privacy preserving delegated access control in the storage as a service model,” IEEE International Conference on Information Reuse and Integration (IRI), 2012.

[9] L. Su, D. W. Chadwick, A. Basden, and J. A. Cunningham. Automated decomposition of access control policies. In Proc. of International Workshop on Policies for Distributed Systems and Networks, pages 3–13, 2005.

[10] Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo, "Policy Decomposition for Collaborative Access Control", ACM Symposium on Access Control Models and Technologies(SACMAT), Colorado, USA, 2008.

[11] Collaboration, http://en.wikipedia.org/wiki/Collaboration, April 26, 2014.

[12] Why collaboration is Crucial to success, http://www.fastcompany.com/3024246/leadership-now/why-collaboration-is-crucial-to-success, April 26, 2014.

[13] Cannabis Companies Garner Attention As Sector Continues to Grow: Company Announces Collaboration To Develop Seed To Sale Software, http://www.marketwatch.com/story/cannabis-companies-garner-attention-as-sector-continues-to-grow-company-announces-collaboration-to-develop-seed-to-sale-software-2014-04-10, April 26, 2014.

[14] Wave Systems Corp. Announces Expanded collaboration with Micron Technology, http://ustradevoice.com/wave-systems-corp-nasdaqwavx-announces-expanded-collaboration-with-micron-technology-inc-nasdaqmu-3606.html, April 26, 2014.

[15] 50 examples of business collaboration, http://www.co-society.com/wp-content/uploads/CO_business_2013.pdf, April 26, 2014.

[16] Benefits of Cloud Computing: Tools that Improve Communication & Collaboration, http://sheepdog.com/2013/04/benefits-of-cloud-computing-tools-that-improve-communication-collaboration/, April 26, 2014.

[17] Event Industry Veteran Launches Start-up, EventCollab - Cloud-based Collaboration Software That Will Make Event Professionals More Efficient, http://www.telepresenceoptions.com/2014/04/event_industry_veteran_launche/, April 26, 2014.

[18] Oracle Teams With Amazon, Intel in Cloud-Seeding Deals,http://www.theregister.co.uk/2011/09/02/icloud_runs_on_microsoft_azure_and_amazon/, April 26, 2014.

[19] H. Walaika. 2008. Retrieved April 26, 2014 from http://www.technewsworld.com/story/64617.html.

[20] Oracle and Microsoft announce cloud collaboration, http://www.businesscloudnews.com/2013/06/25/oracle-and-microsoft-announce-cloud-collaboration/, April 26, 2014.

[21] Cloud Computing Collaboration: Appirio Marries Salesforce Chatter, Google Apps, http://www.crn.com/news/cloud/228600161/cloud-computing-collaboration-appirio-marries-salesforce-chatter-google-apps.htm, April 26, 2014.

[22] Salesforce.com, HP unveil 'Superpod' cloud collaboration, http://www.zdnet.com/salesforce-com-hp-unveil-superpod-cloud-collaboration-7000023334/, April 26, 2014.

[23] Amazon Rivals Collaborate: Dell, Microsoft Unveil Cloud Partner Ecosystems, http://www.datacenterknowledge.com/archives/2013/12/12/amazon-rivals-collaborate-dell-microsoft-unveil-cloud-partner-ecosystems/, April 26, 2014.

[24] Collaborating Clouds, http://nasawatch.com/archives/2010/06/collaborating-c.html, April 26, 2014.

[25] Cloud Computing. 2014, http://en.wikipedia.org/wiki/Cloud_computing, April 26, 2014.

[26] Security Guidance from Critical Areas of Focus in Cloud Computing. https://cloudsecurityalliance.org/csaguide.pdf, April 26, 2014

[27] K. L. Ronald, V. D. Russell, Cloud Security, A Comprehensive Guide to Secure Cloud computing. Indianapolis: Wiley Publishing Inc., 2010.

[28] Levenshtein distance, http://en.wikipedia.org/wiki/Levenshtein_distance, April 26, 2014.

[29] Disjunctive Normal Form, http://en.wikipedia.org/wiki/Disjunctive_normal_form, April 27, 2014.

VITA

Pavani Gorantla was born in India. She earned her bachelor’s degree in Computer Science Engineering from Jawaharlal Nehru Technological University, India in June 2012. She has been a graduate student in Computer Science Department at Missouri University of Science and Technology from August 2012 and worked as a graduate research assistant under Dr. Dan Lin. She received her Master’s degree in August 2014. She has a relevant internship experience as an Operations Research intern during her Master’s program.