Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 5 (2017) pp. 025-032
© MIR Labs, www.mirlabs.net/jnic/index.html

Inter-Operability In Incompatible Access Control Models

Kanthi Kiran Bhargav P.V
Amity School of Engineering and Technology,
Noida, Uttar Pradesh.
arjhnpr@gmail.com

Deepti Mehrotra
Amity School of Engineering and Technology,
Noida, Uttar Pradesh.
dmehrotra@amity.edu

***Abstract*--Sharing of resources across an interoperable environment has it's own fair share of pros and cons. The sharing of resources or data in an interoperable environment enhances the quality of services and the productivity among the organizations. The pros of an interoperable environment are mitigated by it's cons. The resources and data of the organisations are under constant threat of being accessed beyond the authorization privileges specified to them. In an intra-operable environment, these threats of accessing beyond the privileges specified are nullified by defining Security Policies (SP) and implementing these defined Security Policies (SP) through Access Control Models. In an inter-operable environment, defining Security Policies (SP) for the whole environment can be a tedious and a complex task. This approach makes the system a rigid one and performs poorly while handling a large number of users/entities i.e it is not scalable. This approach will be more complex when both the organisations employ incompatible Access Control Models. A more effective approach would be to design a system which is efficient enough to make the Access Control Models in the environment compatible with each other. This can be achieved by designing a system which can dynamically generate Access Control Policies (ACPs) for a specific Access Control Model, which are compatible with the remaining Access Control Models in the environment. In this paper we focus on the inter-operability of two such models - Attribute Based Access Control Model (ABAC) and Role Based Access Control Model (RBAC).**

Keywords: Role Based Access Control Model, Attribute Based Access Control Model, eXtensible Access Control Markup Language, Policy Segregator,Policy Generator

I.Introduction

The hierarchical distribution of authority privileges among the entities of an organisation tends to get complex with the growth of the organization over time. The Access Control Models (ACMs) are employed in order to ease the process of distribution of authority privileges among the various entities of the organisation. The strategies for these distribution of authority privileges among the entities of the organization evolved from the rigid approach of manually assigning the privileges to the Users (ACLs) as defined in [1] to the dynamic approach of deducing the extent of privileges based on the attributes assigned to the users and resources [2]. Though there has been progress in the intra-operable environment, the arena of inter-operable environment is still unexplored.

The organizations spend a huge amount of time and monetary resources for the sole purpose of securing their resources and data. The Access Control Models (ACM) defined for the intra-operable environment makes sure that the data and resources of the organization are secure by restricting the access privileges to the entities based on their ranks and levels. In a scenario of inter-organizational sharing of resources, there may arise a situation where incompatible Access Control Models are employed; due to the lack of proper framework or a system present for inter-operable environment, the resources and the data of the organisations are at risk of being accessed beyond the extent of sharing agreed to by the organisations. In this paper, the study is being conducted on the inter-operability among the incompatible models which are defined using the same XACML framework employed in [3] [4]. A case study is considered, where two organizations implement two different Access Control Models – where one implements the Role Based Access Control Model and the other implements the Attribute Based Access Control Models.

This paper is organized as follows: Section 2 provides a background study of the Access Control Models, Access Control Models employed and the XACML framework employed in this paper. The Section 3 proposes a model for making the Role Based Access Control Models (RBAC) compatible with the Attribute Based Access Control Models (ABAC). The Section 4 implements the model proposed mentioned in the Section 3 of the paper. The Section 5 consists of the issues which are existent in this system and which can be rectified in the future. This is followed by Conclusion and Result in Section 6 and the Future work in Section 7.

II.Background Study

Access Control Models (ACM) are models which are used to define the distribution of authorization privileges among various entities in the organisation. The main component of Access Control Models is represented by it's security policies that regulate the access to data and resources as cited in [17]. The Security Policies of the Access Control Systems in turn consists of Policy Sets which are constituted by a set of Policy Rules. The administrator is responsible for framing these security policies, keeping in mind the need for access control among the entities [5].

An access request can be defined below (1),

$$S,A,R,(condition) \rightarrow auth.(User,\ Action,Request)$$

(S ε Subject Set, A ε Activity Set, R ε Resource) (1)

Whenever a user (subject) submits a request to access the resources i.e the access request, the Authorization System compares the access request with the security policies defined in the system. If the result turns out to be a positive one, then the user is authorized to access the resource/data entity or else the authorization is declined. This is depicted in the figure 1 below,

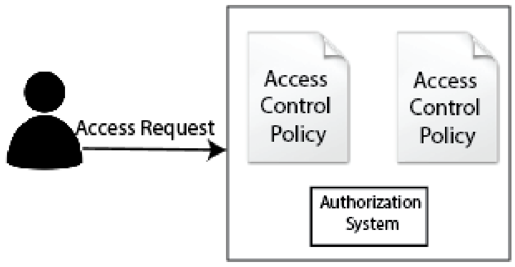


Fig 1. An overview of ACM

This is the general approach employed by various Access Control Models

A. Role Based Access Control Model (RBAC) **[7]**

Role Based Access Control Model classifies the user into groups called roles, which in turn are allotted permissions, unlike the other Access Control Models where the permissions are assigned to the user. This approach of assigning the permissions to the groups instead of users, tends to make the system more flexible, where a new user can be assigned permissions without any hassles. The basic components of a Role Based Access Control Model are:

1) *Users (U)*: The users are the subjects of this Access Control Model. The user can be a monotonic user like a human being or an automated system. The user is in turn allotted to one or many roles.

2) *Roles (R):* The role is a group which are assigned the necessary permissions required to fulfil the responsibilities of the role. The roles and permission have a one-to-many relationship among them.

3) *Permissions (P):* The permissions are positive authorizations to access a resource/data entity. The negative authorizations aren't defined under the permissions set, they are implemented through constraints[12].

4) *Sessions (S):* The sessions are used to keep track of the activity of the subject. A session is initiated whenever a role set is activated by the user. A user can have multiple sessions at a single point of time.

B. Attribute Based Access Control Model (ABAC) [8]

The Attribute Based Access Control Model (ABAC) implements the Subject (SA), Resources (RA) and Permissions (Actions) through attributes. This implementation of the basic components of an Access Control Models (ACM) through attributes coupled with Environment Attributes (E.A) tends to make the model more dynamic. The basic components of an Attribute Based Access Control Model are:

1) *Attributes:* The attributes are the variables which define a resource or a subject. The attributes can vary from being a set of distinct and non-linear values to a set of a range of values. The attributes in Attribute Based Access Control Model (ABAC) can be classified as Environmental Attributes, Subject Attributes and Resource Attributes.

2) *Actions:* The Actions are used by the user to access/change a resource. The various actions which are available for a user of the highest level are read, write, append and delete.

3) *Policy Set:* The Policy Set is a combination of Policies and Policy set. The Policy Combining Algorithms (PCA) are implemented in a policy set to deduce a result out of two conflicting results. While on the lower level, the Rule Combining Algorithms (RCA) are implemented to deduce a result out of two conflicting rules, which in collection make up policies.

C. Extensible Access Control Markup Language (XACML)

The Extensible Access Control Markup Language is an OASIS standard which is used to define security policies as cited in [3]. It is an extension of the Extensible Markup Language (XML), but with a pre-defined schema to define the attributes and rules.

XACML provides features which support a wide variety of Policies. It provides standardized syntax for requesting an action from a system. The Actions can classified into the following types [16]:

1) *Permit* – This permits the system/user to perform an action i.e positive outcome.

2) *Deny* – This denies the system to perform an action i.e negative outcome.

3) *Intermediate* – This is raised when an error or an incorrect values prevents the action from taking place.

4) *Not Applicable* – This is raised when the request can't be processed.

A basic Access Request can be implemented using the following attributes and tags:

1) *Subject:* The <Subject></Subject> tag can be used to define the subject along with it's attributes.

2) *Resource:* The <Resource></Resource> tag can used to define the resource and it's attributes.

3) *Action:* The <Action></Action> tag can be used to define the type of action.

D. *Implementation of Security Policies in RBAC* [3]

The Role Based Access Control Model (RBAC) is implemented using XACML. A complex approach is employed while the framing Security Policies of Role Based Access Control Model (RBAC). The various types of Policy Sets employed while defining RBAC security policies are:

1) *Role Policy Set (RPS):* The Role Policy Set defines the various users/subject associated with the role. The Role Policy Set holds the attribute values of the roles. It contains the reference to it's respective Permission Policy Set. The RPS is a subset of the sets of User and Roles.

2) *Permission Policy Set (PPS):* The Permission Policy Set defines the various permissions associated with the role. The PPS is a subset of the Permissions set.

3) *Separation of Duties Policy Set (SDPS):* The Separation of Duties policy set is used in preventing a user having conflicting roles from accessing a resource. The constraints are defined in this policy set.

4) *Role Assignment Set (RAS):* The Role Assignment Set is a mitigation approach to the Separation of Duties Policy Set, which prevents the administrator from assigning conflicting roles to the user.

E. *Implementation of Security Policies in ABAC* [4]

The XACML is the most perfect fit for defining the security policies of the Attribute Based Access Control Model (ABAC). The various attributes in Attribute Based Access Control Model are:

1) *Subject Attributes (SA):* The Subject attributes are used to define the attributes of the user/subject.

2) *Resource Attributes (RA):* The Resource Attributes are used to define the attribute of the user/subject.

3) *Environmental Attributes (EA):* The Environmental Attributes are what make the ABAC system dynamic, they can be used to define the extent of authority based on the environment i.e time, place or situation.

The Subject set in an ABAC model comprises of the sets of Subject Attributes and Environmental Attributes, while the action is defined by the user/subject which belongs to the activity set.

III. PROPOSED THEORY

The Attribute Based Access Control and Role Based Access Control Models may vary in their approach while defining their respective Security Policies, but they both tend to follow the basics of implementing an Access Control Model. The models might have different approaches, like every other Access Control Model, they tend to use a similar approach for authorization i.e Access Request (AR).

The Role Based Access Control Models differ from the common Access Control Models as they don't share a common ancestor as cited in [14]. This can be mitigated by combining all the Policy Sets defined for a Subject or a role.

A paper states that "RBAC defines roles between users and permissions, ABAC defines attributes that can be required or forbidden in order to give users access to resources" [15].

The Attribute Based Access Control Model has a single Policy Set to define it's Security Policy while the Role Based Access Control Model tends to employ three policy sets for defining it's Security Policy. This is depicted in the following figure 2 below,

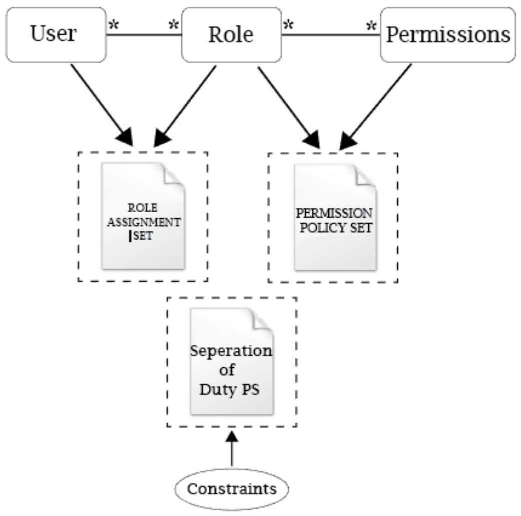


Fig 2. Mapping of RBAC with it's implementation

The Role Based Access Control Model's Policy Sets aren't compatible with that of the Attribute Based Access Control Model. Unlike those of ABAC, the Policy Sets don't necessarily define a single subject. The Role Assignment Set (RAS) can consists of more than one Subject assigned to a single role. The Permission Policy Set (PPS) can consists of more than one permissions defined for accessing the resources associated with the Role Based Access Control Models. This multiple declaration of subjects or permissions associated with the resources can render the Policy Set incompatible with the Policy Sets of the Attribute Based Access Control Models (ABAC)

The Proposed Model would be to define a system which maps the Subjects to the Roles and in turn maps the Roles to the Resources. The Subjects, the Resources and the Permissions (actions) from the various Policy Sets are grouped to form Rules Set for the ABAC Policies. The occurrence of multiple Subjects or multiple resources and their associated actions are handled by mapping the Subjects, Permissions and Resources repetitively i.e different set of rules are framed for the different Subjects of the RAS Policy Set.

The mapping of the Subjects, Resources and Permissions can be achieved by interpreting the information about the Subjects or Resources or Permissions from the respective Policy Sets. The interpreted information has to be used in reframing new rules and which in turn make up a new Policy Set. The major challenge with this approach is maintaining the XACML Schema in the new Policy Set. This can be mitigated to an extent by manually defining the schema for the new Policy Sets.

The proposed system consists of two phases – the necessary information is interpreted in the first phase while a new Policy Set is framed out of the interpreted information in the second phase. The Proposed model would consists of two modules – a Policy Segregator and a Policy Generator, one for each phase. The Policy Segregator would interpret the necessary information by parsing the XACML Policy Sets, while the Policy Generator would generated a Policy Set out of the Parsed data. This is depicted in the figure 3 below,

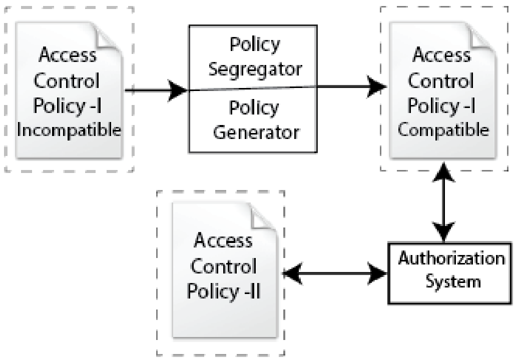

Access Control Policy -I Incompatible
Policy Segregator
Policy Generator
Access Control Policy -I Compatible
Access Control Policy -II
Authorization System

Fig 3. The Proposed Model

In Simple terms, the Proposed Model would amalgamate the necessary information obtained from the various Policy Sets- Role Assignment Set, Separation of Duties Policy Set and Permission Policy Set into a resultant Policy Set which is compatible with the ABAC Policy Sets. This is depicted in the figure 4 below.

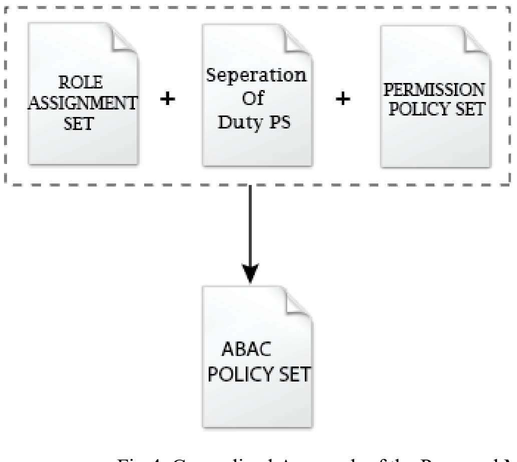


Fig 4. Generalized Approach of the Proposed Model

IV. Implementation

The system proposed consists of two modules- Policy Segregator and Policy Generator. This division of the system into two modules rather than one module reduces the complexity of the system. The division of the system into two modules makes the effective reuse and minimization of code. These modules can be implemented using a Python script. The main objective of these modules is to parse the Access Control Policies, extract the necessary information out of the Access Control Policies and generate a new Access Control Policy which would be in compatible with the other model in the environment. The Python Script utilizes a XML parser – an Element Tree XML API or a Document Object Model (DOM) API to parse the XACML policies to generate the information or create new XACML policies.

The two models which are employed in this paper are – Role Based Access Control Model and Attribute Based Access. The next step would be choosing the model which would serve as an input for the system. The Role Based Access Control Model is the one which is preferred as it is an outdated model compared to the Attribute Based Access Control Model (ABAC). The main challenge with Role Based Access Control Model (RBAC) is that there’s always a stronger connection between strong security and easier administration [16].The security and the administration have an inverse relation, thus striking a balance between both is a complicated issue. The Attribute Based Access Control Model (ABAC) is more granular and more flexible. The major con which outweighs the Role Based Access Control Models (RBAC) is that the Attribute Based Access Control Model (ABAC) is a more dynamic model.

The two modules proposed for the system are:

1) *Policy Segregator:* The main objective of this Module is to parse the XACML Policy, extract the necessary information from the Policy Sets. This module parses four Policy Sets- Permission Policy Set, Role Policy Set, Role Assignment Set and Separation of Duties Set. The information extracted in this module is store in variables and passed onto the next module. The Flow of control of this module is depicted in the Figure 5.

2) *Policy Generator:* The main objective of this module is to create a new Policy Set which is compatible with the Attribute Based Access Control Model (ABAC). The information passed on to by the Policy Segregator Module is collected and Policy Sets are framed out of the passed on parameters. An XACML schema is pre-defined in this module, which is employed while defining the new Policy Set. The Flow of control for this module is depicted in the figure 6.

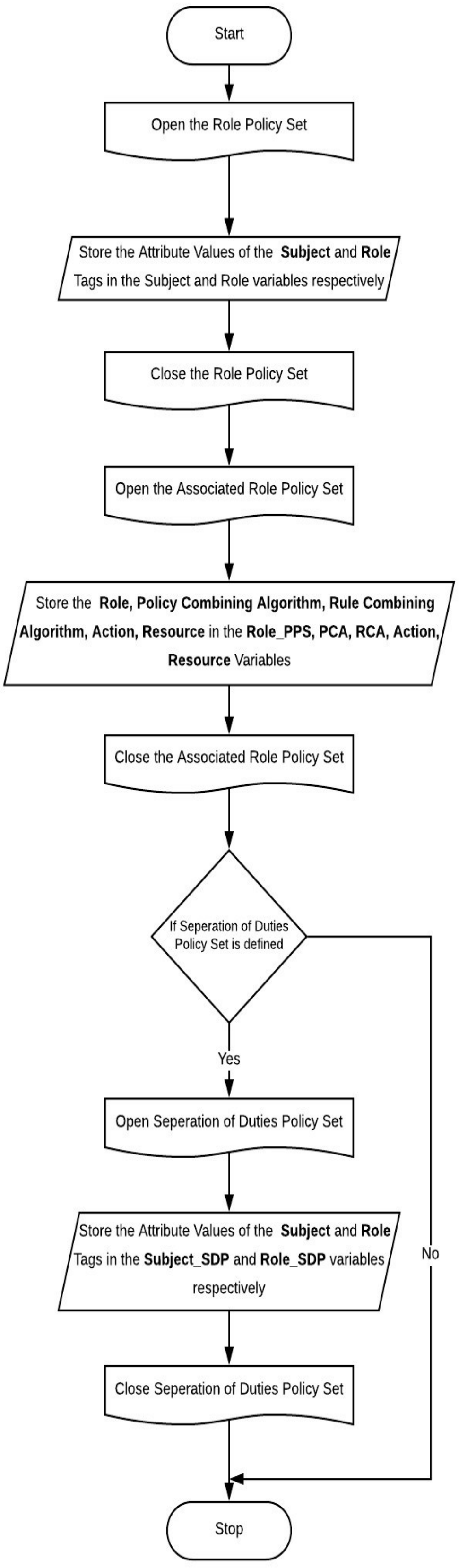

Start
Open the Role Policy Set
Store the Attribute Values of the **Subject** and **Role** Tags in the Subject and Role variables respectively
Close the Role Policy Set
Open the Associated Role Policy Set
Store the **Role, Policy Combining Algorithm, Rule Combining Algorithm, Action, Resource** in the **Role_PPS, PCA, RCA, Action, Resource** Variables
Close the Associated Role Policy Set
If Seperation of Duties Policy Set is defined
Yes
No
Open Seperation of Duties Policy Set
Store the Attribute Values of the **Subject** and **Role** Tags in the **Subject_SDP** and **Role_SDP** variables respectively
Close Seperation of Duties Policy Set
Stop

Fig 5. Flow of Control for Segregation of Attributes

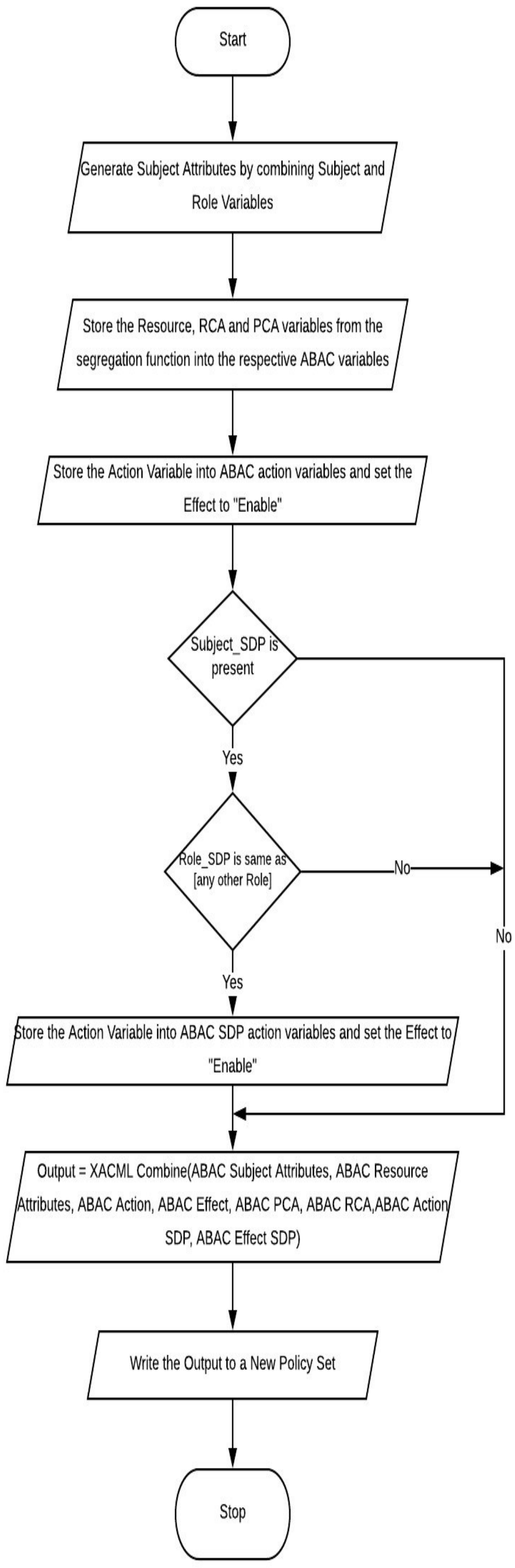

Start
Generate Subject Attributes by combining Subject and Role Variables
Store the Resource, RCA and PCA variables from the segregation function into the respective ABAC variables
Store the Action Variable into ABAC action variables and set the Effect to "Enable"
Subject_SDP is present
Yes
No
Role_SDP is same as [any other Role]
No
Yes
Store the Action Variable into ABAC SDP action variables and set the Effect to "Enable"
Output = XACML Combine(ABAC Subject Attributes, ABAC Resource Attributes, ABAC Action, ABAC Effect, ABAC PCA, ABAC RCA,ABAC Action SDP, ABAC Effect SDP)
Write the Output to a New Policy Set
Stop

Fig 6. Flow of Control for Generation of Policy

The first module of the proposed system is the Policy Segregation module. The algorithm for the process of Policy Segregation is defined below. The Policy Sets defined for the RBAC model serve as inputs for this algorithm.

1) Open the Role Assignment Set (RPS) from the list of policy sets.
2) Parse the Role Assignment Set, until the cursor comes across the Subject Tag. Store the Attribute Value of the Subject field into a variable
   Subject = Attribute Value of the Subject Tag.
3) Parse the Role Assignment Set, until the cursor comes across the Resource Tag. Store the Attribute Value of the Associated Role field into a variable
   Role = Attribute Value of the associated Role Tag
4) Parse the Policy Set, when you come across the Action tag, store the Attribute Value of the Action tag into a variable.
   Effect = Attribute Value of the Action Tag
5) Close the Role Assignment Set, once the parsing of the Policy Set is done.
6) Open the Role Policy Set from the list of Policy Sets
7) Parse the Policy Set, when you come across the PPS tag, store the Attribute Value of the PPS tag into a variable. The PPS name is essential as it links the Permission Policy Set with it's Role Policy Set.
   PPS_Name = The name of the associated Permission Policy Set (PPS)
8) Close the Role Policy Set (PPS), once the Subject, Role and PPS tag values are stored.
9) Open the Permission Policy Set. The Permission Policy Set name will be store in the PPS_Name variable
10) Parse the Permission Policy Set, when the cursor comes across Role Tag, store the attribute value into a variable.
    Role_PPS = Role associated with the Permission Policy Set (PPS)
11) Parse the Permission Policy Set for the Policy Combination Algorithm and store it's value.
    PCA= The Policy Combination Algorithm employed
12) Parse the Permission Policy Set for the Rule Combination Algorithm and store it's value.
    RCA= The Rule Combining Algorithm employed
13) Parse the Permission Policy Set for the Action assigned to the Role and store it's value.
    Action = Value of the Action associated with the Permission Policy Set (PPS)
14) Parse the Permission Policy Set for the Resource which the role has to perform it's responsibilities and store it's value.
    Resource = Value of the Resource associated with the Role Policy Set (RPS)
15) Close the Permission Policy Set, once the Parsing is completed.
16) If the Seperation of Duties Policy Set is defined for the role then,
    16.1) Open the Seperation of Duties Policy Set, this Policy Set contains the constraints associated with the user/subject
    16.2) While parsing the Seperation of Duties Policy Set, the Subject defined in the Policy Set needs to be stored.
    Subject_SDP = Attribute Value of the Subject Tag
    16.3) While parsing further, the role which the user is forbidden from being assigned needs to be stored.
    Role_SDP = Attribute Value of the Role Tag
    16.4) Close Seperation of Duties Policy Set once the parsing of the Policy Set is completed.

The second module of the proposed system is the policy generation module. The input for this module is the output parameters passed by the Policy Segregation Module. The algorithm for the process of Policy Generation is defined below,

1) Once the Policy Segregation Module is completed, the Policy Generation Module is called for.
2) The first step would be to pass the Subject and Role variables obtained from the Policy Segregation Module into a Combine Function.
   ABAC Subject Attributes = Combine(Subject, Role)
3) The resource variable are used to generate the Resource Attributes.
   ABAC Resource Attributes = Resource
4) The Action variable is store in the ABAC action.
   ABAC Action = Action
5) The Effect is set to "Enable" or "Disable" as obtained from the Role Assignment Set.
   ABAC Effect = Effect
6) The Policy Combination Algorithm for the compatible Policy Set is assigned the same Policy Combination Algorithm of the previous Access Control Model.
   ABAC PCA = PCA
7) The Rule Combination Algorithm for the compatible Policy Set is assigned the same Rule Combination Algorithm of the previous Access Control Model.
   ABAC RCA = RCA
8) The Action variable for the Seperation of Duties Policy Set is initialized to Null.
   ABAC Action SDP = NULL
9) The Effect variable for the Seperation of Duties Policy Set is initialized to Null.
   ABAC Effect SDP= NULL
10) The next step would be to check whether the Subject_SDP variable contains a Null value or it contains a value. If the Subject_SDP value is not equal to Null, the system gets to know that the Subject has constraints.

If Subject_SDP is present:
10.1) If the Role_SDP value is not equal to the Role value , then
If Role_SDP is not same as [Role]
10.2) The constraint Action is stored into a variable.
ABAC Action SDP = Action
10.3) The constraint Effect is set to Deny
ABAC Effect SDP= "Deny"

11) The Output is generated by passing all the arguments obtained into the XACML_Combine function which generates the Policy Set in an XACML format.
Output = XACML Combine(ABAC Subject Attributes, ABAC Resource Attributes, ABAC Action, ABAC Effect, ABAC PCA, ABAC RCA,ABAC Action SDP, ABAC Effect SDP)

12) Write the Output returned by the XACML Combine function into a New Security Policy.

The following functions are defined in the Policy Generator Module:

1) *Combine(<Parameters>):* The Combine function is a user-defined function which appends it's parameters to a parent. The parent in the Policy Generator Module is the Subject. While implementing this function, the Element Tree XML API [13] is used.

2) *XACML Combine(<Parameters>):* The XACML Combine is a user defined function, which appends all it's parameters, along with a predefined XACML schema to the root node i.e the XACML Header. This function also employs the Element Tree XML API[13]

V. Discussion

The Approach presented in this paper was successfully evaluated against the objective initially defined. There are other relevant issues which need to discussed, but don't necessarily form the key issues. These issues are:

1) *Scalability:* The System defined here is a static system. Thus, it isn't scalable. The Future scope would be to design a dynamic system which would be scalable.
2) *Autonomic Computing:* The System is fully autonomic, it parses the XACML policies, without the intervention of any user.
3) *Portability:* The System designed is portable on any environment, if all it's dependencies are met.
4) *Compatibility:* The System designed her is compatible with only two Access Control Models. The Future Scope would be to extend it to various other models.

VI. Conclusion And Result

The Algorithm defined above is for the interoperability of Role Based Access Control Models and Attribute Based Access Control Models. The proposed model can be implemented using a XML parser and a python script which can be used to define the Policy Segregator and the Policy Generator.

A. *Case Study*

For a Case Study, we would take into consideration, a subject name "Manager". Manager is assigned a role of a Manager, and he would have four Policy Sets accordingly- Role Policy Set, Permission Policy Set , Role Assignment Set and Separation of Duties Set.

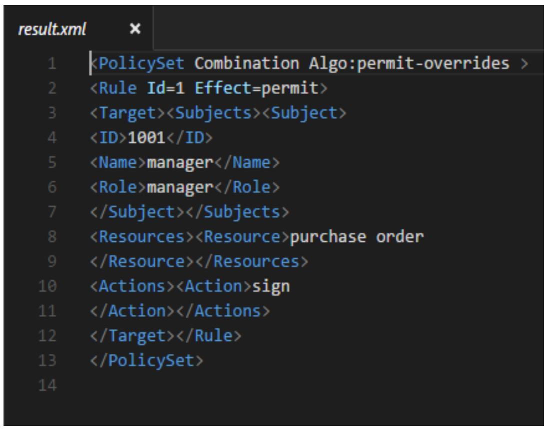
The First Step would be to parse the Role Assignment Policy, which would produce the Subject i.e Manager, Role i.e Manager and the Effect ="Permit"

The Second Step would be to parse the Role Policy Set, it would yield the Permission Policy Set's name.

The third step would be to parse the Permission Policy Set, it would yield RCA="Permit Override", PCA="Deny Override", Action="sign" and Resource="Purchase Order".

The fourth step would be to check whether a Separation of Duties Policy Set is defined, as there is no Separation of Duties Policy Set, it skips the step.

The Fifth Step, the Policy Generator Module generates a new XACML file called result.xml is created. The following is depicted in the figure 5 below


result.xml

```
<PolicySet Combination Algo:permit-overrides >
<Rule Id=1 Effect=permit>
<Target><Subjects><Subject>
<ID>1001</ID>
<Name>manager</Name>
<Role>manager</Role>
</Subject></Subjects>
<Resources><Resource>purchase order
</Resource></Resources>
<Actions><Action>sign
</Action></Actions>
</Target></Rule>
</PolicySet>
```

Fig 5. Sample Output

VII. Future Work

The need for sharing resources over a network is of great importance for the Organizations, it paves way for the possibility of effective outsourcing of necessary resources. The future scope would be to extend this framework to various other Access Control Models and to design a system which would be compatible with all the Access Control Models present in the environment and would dynamically generate the authorization outcomes based on the access request generated employing the models as stated in [9] [10][11].

References

[1] Romauld Thion, "Access Control Models", Chapter-37, IGI Global Publication", pp:1-4,2008

[2] RajaniKanthi Aluvalu, Lakshmi Muddana,"A dynamic Attribute-Based Risk Aware Access Control Model (DA-RAAC) for cloud computing", IEEE International Conference on Computational Intelligence and Computing Research,2016

[3] Anne Anderson, "XACML Profile for Role Based Access Control",OASIS Open, 2004

[4] Apurva Mohan, Douglas M.Blough, "An Attribute-based Authorization policy Framework wth Dynamic Conflict Resolution", IDtrust'11, Gaithersburg, pp: 37-50, April,2010.

[5] Bill Fisher, Norm Brickman, Santos Jha, Sarah Weeks, Ted Kolovos, Prescott Burden, "Attribute Based Access Control", NIST Special Publication,2016.

[6] Kamel Adi, Yacine Bouzida, Ikhlass Hattak, Luigi Logrippo, and Sergie Mankovskii "Typing for Conflict Detection in Access Control Policies", 4th International Conference, MCETECH 2009, pp : 212-226.

[7] Ravi S.Sandhu, Edward J.Coyne ," Role-Based Access Control Models ", IEEE Volume:29 Issue:2, pp-38-47 , 1996.

[8] Apurva Mohan, Douglas M.Blough, "An Attribute-based Authorization policy Framework wth Dynamic Conflict Resolution", IDtrust'11, Gaithersburg, pp: 37-50, April,2010.

[9] Shalini Bhartiya, Deepti Mehrotra, "Threats and Challenges to Security of Electronic Health Records", in Social Informatics and Telecommunications Engineering 2014, LNICS, Volume 0115, ISBN: 978-1- 936968-71- 8.

[10] Khaled Riad, Zhu Yan, Hongxin Hu, and Gail-Joon Ahn, "AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing", IEEE Conference on Collaboration and Internet Computing,2015

[11] Shalini Bhartiya, Deepti Mehrotra, Anup Girdhar, "Proposing hierarchy-similarity based access control framework: A multilevel Electronic Health Record data sharing approach for interoperable environment", Journal of King Saud University - Computer and Information Sciences, 2016

[12] David F.Ferraiolo , D.Richard Kuhn., " Role Based Access Controls", 15th National Computer Conference, Baltimore MD, pp. 554-563,1992.

[13] Element Tree XML API – https://docs.python.org/2/library/xml.etree.elementtree.html

[14] Lionel Montrieux,Michel Wermelinger, Yijun Yu, "Challenges in Model-Based Evolution and Merging of Access Control Policies", Joint 12th International Workshop on Principles on Software Evolution and 7th ERCIM Workshop on Software Evolution, 5-6 Sep 2011, Szeged, Hungary.

[15] Lionel Montrieux , "Model-Based Analysis of Role-Based Access Control", thesis submitted to The Open University, May,2013.

[16] Vincent C.Hu, David F.Ferraiolo, D. Rick Kuhn, "Assessment of Access Control Systems", National Institute of Standards and Technology Interagency Report 7316, 2006

[17] Federica Paci, Anna Squicciarini, and Nicola Zannone. "Survey on Access Control for CommunityCentered Collaborative Systems". ACM Comput. Surv. 51, 1, Article 6 (January 2018), 38 pages.