Comparative analysis of Role Base and Attribute Base Access Control Model in Semantic Web

Sonu Verma
Department of CSE.
A.I.C.T.R., GGSIPU
Delhi, India

Manjeet Singh
Associate Professor
Y.M.C.A. U.S. & T.
Faridabad, India

Suresh Kumar
Assistant Professor
A.I.C.T.R., GGSIPU
Delhi, India

ABSTRACT
To control the unauthorized access to the resources and services is an emerging security issue in Semantic Web (SW). There are various existing access control models such as Role base, Attribute base, Credential base, Concept level access control models. They all have some strengths and weaknesses with them. In this paper we first take an overview of history of access control models and the need of access control models in semantic web. This paper has a discussion of strengths and weaknesses of the RBAC and ABAC. Than we have a comparative analysis of RBAC and ABAC with some points of issue.

General Terms
Access Control Model in Semantic Web

Keywords
Access Control Model, Semantic Web, Authorization

1. INTRODUCTION
The semantic web is the next generation of the current web in which computers can interpret the meaning of web content because of explicit semantics provided in markup. The semantic web components are deployed in the layers of the web technologies and specification. Ontologies are taxonomies which use semantic relationship among terms and attributes as well as strict rules about how to specify terms and relationship.
Ontology is the highest level in semantic web layers which has been completely designed and well defined. Ontology is used to: share common understanding of the structure of information, to make the domain knowledge reusable, to make the explicit domain assumptions, to make the domain and operational knowledge separate and to analyze the domain knowledge. It contains the most important part of the system i.e. data, meaning of data and rules & logic for deduction and decision making. There are some security related issues which are regarding to ontologies are as:

- How much or which part of the data should be accessed and by whom?
- Which queries should be answered and till which depth?
- Who is going to access the information and which portion of it?
- How to know the identity of a person exploring the ontology [1]?

This paper is organized as follows: in section2 we will study about the access control and have an overview of the access control models. In section 3 & 4 we will study the access control models and strengths and weakness with them. In section 5 we have a comparative analysis of the access control models. In section 6 we conclude the paper and the future work with access control models.

2. ACCESS CONTROL
Two parallel themes in access control research are prominent in recent years. One has focused on efforts to develop new access control models to meet the policy needs of real world application domain. In a parallel and almost separate thread researcher have developed policy languages for access control [2].
Access control is the mechanism by which services know whether to honor or deny requests. There are four pieces to the problem.

Identification: Assigning a responsible party for actions. A responsible party may be a person or a non-person entity (NPE), such as a computer or a router. We'll use the term user to cover both cases.
Authentication: The means used to prove the right to use an identity, take on a role, or prove possession of one or more attributes.
Authorization: The means of expressing the access policy by explicitly granting a right.
Access Decision: Using some combination of the other three to decide whether or not a request should be honored [3].

2.1 Access Control Models
In the early days of the mainframe, when timesharing was new, people realized that the biggest need was to prevent one user from interfering with the work of others sharing the machine. They developed an appropriate access control model, one that depended on the identity of the user. Permission to use a system resource, such as a file, was linked to the user's identity. This approach is called Identification Based Access Control (IBAC). IBAC stores permissions in an access matrix, and the IBAC model doesn't include a specification of permissions for changing its entries. That left it to a trusted party, the system administrator, to make all changes. The administrator typically had unfettered rights to access anything within the system as the number of users grew; the burden on the administrator became untenable. That led to the introduction of additional concepts, such as "owner" and "group." It was hard to understand all the permissions that would be granted by adding a user to a group, and the rights granted "owner" did not allow for Mandatory Access Control (MAC).

While IBAC could manage centralized monolithic systems, distributed systems proved to be problematic for

IBAC. Users could have many identities and often had to authenticate in different ways on different systems, which led to work to consolidate access control systems with Federated Identity Management, FIdM, and Single Sign-On/Single Log-Out (SSO/SLO). Managing the access rights for individuals and machines became too large a burden and prone to error, particularly de-synchronization. As complexity increased Role Based Access Control (RBAC) was offered as a solution. Permissions in the access matrix were tied to roles, and which users could assume a particular role became the means of controlling user access [4]. Challenges with RBAC became apparent when it was extended across domains. Reaching agreement with all partners on what rights to associate with a role proved to be difficult. Adding, deleting, or modifying the duties of a role involved updating too many policy stores. Further, RBAC had limited support for context, such as day vs. night or war vs. peace, when it was important in the access decision [4].

Attribute Based Access Control (ABAC, sometimes referred to as Policy Based Access Control or PBAC) or Claims Based Access Control or CBAC), was proposed as a solution to these new issues. The access decision would be based on attributes that the user could prove to have, such as clearance level or citizenship. That approach made it easy to include context in the access decision. As it evolved it was also called Risk Adaptive Access Control (RAdAC). Access implications of changing a user's attributes where left unspecified in most conceptualizations. There is also the issue of reaching agreement on the meaning of attributes when spanning organizations because of the need to reconcile complex and extensive lexicons. [3]

We will study the existing access control models and find out the strength and weaknesses of them. We will do a comparative analysis of the Roll Based Access Control (RBAC) and Attribute Base Access Control (ABAC) models and also take an overview of the strengths and weakness of both upon each other.

3. ROLE BASED ACCESS CONTROL MODEL (RBAC)

In the development of the RBAC ontology, we have followed these principles [5]: (1) the access control ontology should limit itself to expressing the modeling abstractions of RBAC. (2) No hypothesis is made about the domain knowledge, neither semantically nor syntactically the resulting role ontology shall not depend on external factors like the type of organization or type of procedures nor on the structure of the domain ontology (which could be either monolithic or a layered system composed of different ontologies). (3) We do not capture any workflow procedure inside the model so as to preserve both simplicity and generality.

The RBAC model uses the following four classes:
Action. This is a partial, or self-standing, class that represents an action that can be performed by a user on a resource. e.g., Read, Write.
Resource. This is a defined class, representing the authorization objects. The DL classifier will place under Resource all the classes that match the condition.
Privilege. Privileges are the permissions which are granted to the user according to his role e.g. read, write, create, delete etc.
Role. This is a defined class which classifies the user in the domain system. We then expect that any class that is declared to have a privilege in the domain ontology to be classified as Role e.g. Manager, student, library card holder etc.

3.1 Overview of RBAC

A role based access control model is very simple and easy to use. Roles are assigned to user statically by the security administrator but it can be restrictive or problematic in some situations such as mobility and the pervasive computing and it can be restrictive in collaborative environment also which is highly demandable in sharable environment.

Sometimes RBAC model proved inefficient for some reasons, first to differentiate roles in different context proved to be difficult and resulted in large scale of role definition. In some cases it produces more roles than users. On the other hand RBAC not produces fine grained results while requirements are fine grained now days [9].
RBAC assigned the roles to its user statically, which is not preferable in dynamic environment. It is very difficult to change the privilege of the user without changing the role of the user. RBAC is a best access control model for the local domain. It has no complexity due to the static role assignment. The policy specification and maintenance is also need a low attention due to the static nature of roles and the privileges with them.
It becomes problematic when the environment is distributed and dynamic. The RBAC model has no delegation models which are required for the distributed and collaborative environment.

RBAC has been criticized for the difficulty to setting up an initial role structure and for in-flexibility in changing environment. RBAC not provide support for dynamic attributes such as time of the day which is considered to determined user permission. Sometimes to support dynamic attributes in large organizations a "Role explosion" problem can arise, In thousands of separate roles for different collections of permissions [12].

Role models are abstractions of user behavior and duties. These are used to map departments' and members' purview to system resources into system. To manage authority and accessing control in large scale software systems it could be effective to use role concept. RBAC effectively reduces complexity and cost of authority management. We can use role in order to manage members' purview more effectively[11].

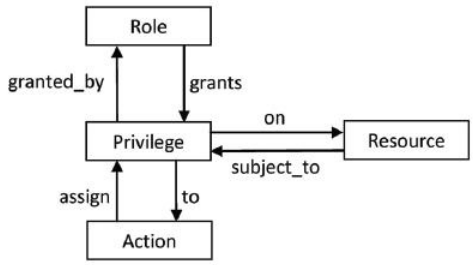

Figure 1: RBAC Model [6]

It establishes relations between users–roles and permissions–roles. However it is difficult to apply the RBAC model when roles cannot be assigned in advance and it is typically not possible to change access rights of a particular entity without modifying the roles. More sophisticated RBAC models allows delegation between roles [2], by delegating the entire set of permissions associated with a set of roles of the delegator to the delegatee. The Figure-1 shows the functionality of a Role Based Access Control Model (RBAC).

4. ATTRIBUTE BASED ACCESS CONTROL MODEL (ABAC)

The basic idea of ABAC is not to define permissions directly between subjects and objects, but instead to use their attributes as the basis for authorizations. For subjects, attributes can be static ones like a subject's name, or position or role in a company. However, dynamic attributes like age, current location or an acquired subscription for a digital library can be used as well. For objects, metadata properties, e.g., the subject of a document can be used. The figure-2 shows the functionality of ABAC model.

Subjects and objects are both represented by a set of attributes and related attribute values. Permissions consist of the combination of a so-called object descriptor, which consists of a set of attributes and conditions like "age > 18" or "subscribed = true", and an operation that is to be executed on the objects denoted by the descriptor. Authorizations are defined between a subject descriptor and permission. Using descriptors it is possible to dynamically assign permissions to subjects and objects, thus making a manual assignment superfluous.

ABAC uses subject, object, environnement attributs. Those may be retrieved from different locations such as local database, trusted third party or directly from a client with a certain access request. Before using these attributes for access control decision the attribute document need to be validated. At least an integrity check of the document and a validation of corresponding trust associated with an attribute document [10].

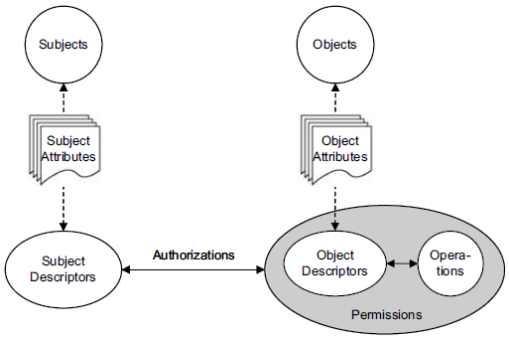

Figure 2: Overview of the ABAC model [7]

For the attribute-based access control techniques at hand, e.g., XACML, all user attributes and conditions to be checked (e.g., "age > 18") must be encoded statically in the policy. Actually, only the conditions necessary from the system designer's point of view (e.g., "fullAge = true") should be relevant when specifying the policies. It is rather a question of attribute management how a user can prove this property.

In ABAC subjects and objects both are represented by the set of attributes and their relative values. The permissions are depending on the combination of these subjects and object attribute values.

4.1 Overview of the ABAC

ABAC overcome the user role assignment problem which exist in RBAC and focuses on the attributes of a user required to grant access. One of those attributes can improve the role a user has assigned inheriting from the RABC model. ABAC is very flexible model that is considerably easier to administer than RBAC [9]. An ABAC works well in an open, distributed and dynamic environment as it has higher flexibility. Higher flexibility comes along with the higher complexity due to the specification and maintenance of the policies. Policy administrators need to be aware of the attribute scheme used by the issuer of a specific attribute, who in many cases may reside in a different organization. The attributes a user possesses do not necessarily match those used by the developers of a web-based information system or service. In classic attribute-based approaches the policy administrators at the different sites have to consider this situation already in advance, which significantly complicates the management of ABAC policies. On the other hand, if the different organizations restrict themselves to a common set of standardized attributes, this would happen at the price of a low expressiveness for the representation of subjects and objects, thus losing some of the advantages of the flexible and dynamic ABAC functionality [7].

An ABAC is very flexible and supportive in a large, open, distributed, sharable and collaborative environment where the numbers of potential users are very high and most of the users are not known before. An ABAC is very supportive in dynamic environment where the roles

of the users are not defined in advance or statically. The subject requests for the objects and permissions are granted based on the subject and object attribute values combination. Authorization of the users is also based on the attributes, which he provides at the time of request. ABAC also has the feature of global agreement such as the user attributes which are provided in one domain are forwarded to the other domain at the time of domain to domain interaction. Sometimes there occurs a problem of mismatching and confusing attributes when attributes which are provided by the user to be authorized and the attributes which are saved in the domain's user attribute database are mismatched due to different formats or versions of the attributes.

On the other hand the heterogeneity of user information increases the complexity of ABAC. This problem can be solved by the centralized database system in which all user attributes are maintained in a user attribute database containing all attributes in the same formats. The centralized user attributes database proved beneficial from another point of view. It increases the flexibility, sharing, privacy and global agreement of the user attributes. It also provides the interoperability among several services provides which can use these attributes data dynamically and can decide upon user rights

Table 1: Strengths & Weakness comparison of RBAC and ABAC

Model	**Strength**	**Weakness**
Attribute Based	1. It is very flexible in a large open system where the numbers of potential users are very high and most users will not be known before.	It has the high complexity due to the specification and maintenance of the policies.
	2. Infrastructure is so built up to support the security of the information and users attributes to perform the authorization and authentication task as well	Sometimes the attributes possess by the user don't necessarily match to those used by the service provider of a web based system or service.
	3. It doesn't use the attributes to define permission directly between subjects and objects but it use the attributes as the basis of authorization	Low expressiveness of the attribute given by the different organization with a common set of standardized attributes.
	4. It gives the global agreements of the attributes so that attributes provided in one domain could be forwarded to the other domain during domain to domain interaction.	The heterogeneity of user information increases the complexity, which can only be solved by the centrally maintained database containing all attributes in same format.
	5. Central storage of user attributes which provides the interoperability and sharing among several service providers which can used these attributes data and can decide upon user rights.	
Role based	1. Simplicity	Don't engage sharable semantic domain models
	2. Easy to use	Don't include the ability to reason over utilities
	3. Best for local domain	Don't have delegation models required by dynamic environment
	4. Static defined roles, lower complexity	Don't Support justification advising and negotiation required for greater autonomy
		It has the mobility problem
		It is not possible to change access rights of a particular entity without modifying the roles.

Table 2: Feature Comparison of RBAC and ABAC

Issue	**RBAC**	**ABAC**
Dynamicity	No	Yes
Global Agreement	No	Yes
Flexibility	No	Yes
Simplicity	Yes	No
Authorization decision	Locally	Globally
Granularity	Low	High
Manageability	Simple	Complex
Trust	Locally	Globally
Confusing deputy	No	Yes
Revocation	No	Yes
Changing privileges	Complex	Simple
Policies specification & Maintenance	Simple	Complex
Role explosion problem	Yes	No

5. COMPARISON ISSUES

The following are comparative discussion of RBAC and ABAC with some point of issues.

1.) **Dynamicity**: A RBAC model has not this feature because the roles and permission with them are assigned to users statically in users domain whether ABAC is supportive in dynamic environment because the user attributes are poses at the time of request and the authorization decisions is made according to the attributes.

2.) **Global Agreement**: RBAC not support the global agreement because the permission to the user roles are assigned in the local domain whether ABAC supports global agreement due to domain to domain interaction and sharable user attribute database.

3.) **Flexibility**: RBAC model is not flexible in the open and distributed environment due to its static nature. But ABAC is much more flexible due to its dynamic nature in an open and distributed system.

4.) **Simplicity**: RBAC is simple and easy to use access control model in which permissions are denoted to the roles statically according to the static /predefined policies. On the other hand ABAC is very complex due to its flexibility, heterogeneity of user attributes, global agreement and sharing. For these features the policy specification and maintenance makes ABAC very complex.

5.) **Authorization decision**: In RBAC authorization decision is in advance at the time of permission role assignment in local domain. But in ABAC authorization decision are made globally according to the user credential provided dynamically. This authorization decision is also forwarded at the time of domain to domain interaction.

6.) **Granularity**: RBAC has low granularity due to its local domain and user has least privileges. On the other hand ABAC has the high degree of granularity due to centralize user attributes database. ABAC has more privileges.

7.) **Manageability**: Manageability in RBAC is simple because the subject has the permission to the action according to his role and the policies are made for the roles actions and permissions but the management of the user's attributes is more complex due to the heterogeneity of the user attributes. There is another problem to protect the user attributes in centralize database in an open and distributed environment.

8.) **Trust:** Trust is an important issue in access control mechanism. In RBAC, trust is highly obtained in local domain. In ABAC trust establishment is more difficult due to the global agreement of the attributes in sharable environment.

9.) **Confusing deputy**: in RBAC there is no issue for the confusing deputy because the security administrator assigned the roles to the user and allots the privilege to them according to their role. In ABAC, sometimes there are confusing attributes which may create confusion to the administrator or service providers. This occurs due to the heterogeneity of the user attributes

10) **Revocation**: Revocation of the user roles are occurs sometimes only when new user joins the domain or there is a need to change the privileges of an existing user. But in ABAC the revocation of the user attributes is needed from time to time. Because the attributes has their validity time after that time attribute get expired.

11) **Changing** Privileges: to change the privilege of a user is very difficult in RBAC system because to change the privilege of a user it is compulsory to change the role of the user. For this there is a need to change the policy specification. On the other hand in ABAC system to change the privilege of a user there is no need to change the user identity or the policy specification as the policies are so defined that a user has the permission according to his/her attributes values.

12) **Policy specification and maintenance**: in a RBAC system policy specification is not much complex in comparison to the ABAC system in which it is very complex. The reason of the complexity are the flexibility, dynamicity, sharing in open and distributed environment, heterogeneity of user attributes, mismatching of attributes, confusing attributes etc. attribute database, object database protection is another reason for this complexity. Sharing the attributes in open system can create the problem of integrity or privacy of the user information and policy defined for this purpose are much more complex.

13) **Role explosion problem:**In RBAC there arise a role explosion problem due to the different type of roles in context of different type of duties and permissions related to these duties.

6. CONCLUSION & FUTURE WORK

In this paper we have studied RBAC and ABAC access control models. Moreover we have listed the strengths and weaknesses of these models. In future we will integrate these two models to get some security requirement and to get an access control model for ontology in semantic web which can overcome the weaknesses of these models .We will integrate the two models RBAC & ABAC in such a way that can overcome the existing problems with RBAC & ABAC and can get a fine grained access control model which is highly demandable in shareable, open and changing environment.

7. REFERENCES

[1] Muhammad Reza Fatemi, Atilla Elçi and Zeki Bayram, "A proposal for Ontology Security Standards", 2008.

[2] Tim Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi Sandhu, William H. Winsborough, Bhavani Thuraisingham, "Using OWL to Model Role Based Access Control", SACMAT'08, June 11–13, 2008.

[3] Alan H. Karp, Harry Haury, Michael H. Davis, "From ABAC to ZBAC: The Evolution of Access Control Models", HP Laboratories-2009-30.

[4] D.F. Ferraiolo and D.R. Kuhn (1992) "Role Based Access Control", 15th National Computer Security Conference, October, 1992.

[5] L. Cirio, I. F. Cruz, and R. Tamassia. "A Role and Attribute Based Access Control System Using Semantic Web Technologies". In International IFIP Workshop on Semantic Web and Web Semantics, volume 4806 of Lecture Notes in Computer Science, pages 1256–1266. Springer, 2007.

[6] Isabel F. Cruz, Rigel Gjomemo, Benjamin Lin, and Mirko Orsini, "A Constraint and Attribute Based Security Framework for Dynamic Role Assignment in Collaborative Environments", 2008.

[7] Torsten Priebe, Wolfgang Dobmeier, Christian Schläger, Nora Kamprath, "Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies", First International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria, April 2006.

[8] A. Mohammad, G. Kanaan, T. Khdour, S. Bani-Ahmad, "Ontology-Based Access Control Model for Semantic Web service", Journal of Information and Computing Science, March, 2011.

[9] Bernard Stepien, Stan Matwin, Amy Felty, "Advantages of a Non-Technical XACML Notation in Role-Based Models", 2011 Ninth Annual International Conference on Privacy, Security and Trust.

[10] Gerald Stermsek, Mark Stremmbeck, Gustaf Neumann, "Using Subject- and Object-specific Attrubutes for Access Control in Web-based Knowledge Management System".

[11] YAO Zhilin, LI Bing and LIU Shufen, "Role Based Collaboration Authorizing by Using Ontology", Chinese Journal of Electronics Vol.20, No.3, July 2011.

[12] D. Richard Kuhn, Edward J. Coyne, Timothy R. Weil, "Adding Attributes to Role-Based Access Control", IEEE Computer Society, JUNE 2010.