Volume 4 No. 5, May 2014 ISSN 2223-4985

International Journal of Information and Communication Technology Research

©2014 ICT Journal. All rights reserved

http://www.esjournals.org

A Security Framework for Access Control in Web Services

Abolfazl Esfandi, Mehdi Sabbari
Department of Computer Engineering Islamic Azad University Borujerd Branch, Iran

ABSTRACT

In this article, we focus on one of the important aspects of Service Oriented Architecture (SOA), which is access control. The article presents the security requirements that must be followed and it presents a conceptual model of requirements in this field based on the needs. Then since different models such as IBAC ،RBAC ،ABAC and RAdAC have been presented so far, we try to present comparison between existing models is presented. After it the ABAC model's structure that is more compatible with SOA is described and there is a comparison between the model and the RBAC model. Since the most important way in implementing SOA is the use of web services, in this article we proposed an architecture for web services in access control to protected services and to adopt some policies on the applications based on ABAC model and SAML standard and XACML languages. The possible activity in the architecture and the implementation stages are explained using use case diagram and sequence diagram in UML.

Keywords: *Service Oriented Architecture, Web Services, Access Control, Security Requirements, RBAC, ABAC.*

1. INTRODUCTION

SOA establishes an architectural model that aims to enhance the efficiency, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing [1]. SOA supplies various services to process business by joining various specific components together in a loose coupled manner with uniform interfaces [2] [5]. SOA and the corresponding Service-Oriented Computing (SOC) have received significant attention recently as major computer and software companies, such as IBM, Intel, Microsoft, HP, SAP, and Sun Microsystems, as well as government agencies, such as U.S. Department of Defense (DoD), have embraced SOA/SOC [2].

Web services seem to become the preferred implementation technology for realizing the SOA promise of maximum service sharing, reuse, and interoperability [3] [4]. Web Service supports the communication between applications developed on different platform by different programming languages with different technological standards. The ultimate goal of Web Service is to realize the integration and interaction between various systems in Internet/Intranet environment, just similar to the function of components [2]. Web services technology platform is comprised of the following core open technologies and specifications: Web Services Description Language (WSDL), XML Schema Definition Language (XSD), SOAP (formerly the Simple Object Access Protocol), UDDI (Universal Description, Discovery, and Integration), and the WS-I Basic Profile [1] [14].

Access control security is one of the important aspects in SOA that is considered as a challenge. This issue requires further attention and review because of the architecture's distributed nature, its high re-usability, simple accessibility and the autonomy of logical solutions units.

There are currently several works that focus on authorization techniques. For instance, eXtensible Access Control Markup Language (XACML) is a popular standard language used to describe the architecture of an authorization policy based on the attributes of the subject, the resource and the environment. Despite its status as an established language, XACML is still undergoing extensive research in both the academic and industrial sectors.

Security Assertion Markup Language (SAML) is another authentication and authorization language. It is designed solely for solving the problem of Web Browser Single Sign-On (SSO), and is therefore not suitable for all authorization situations. A number of models such as (IBAC, RBAC, ABAC, and RAdAC) have been developed to address various aspects of access control problem [11].

Identification Based Access Control (IBAC) stores permissions in an access matrix, and the IBAC model doesn't include a specification of permissions for changing its entries. 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 and Single Sign-On/Single Log-Out (SSO/SLO) [20].

Role-Based Access Control (RBAC) is designed for the task of coordinating users, objects and permissions. However, this model was rendered obsolete with the advent of modern Web technologies, as it could not handle complex authorization tasks.

Subsequently, an Attribute-Based Access Control (ABAC) was designed to overcome the disadvantages of RBAC and to correspond with the architecture of Web Services. ABAC is based mainly on defining the attributes of the entities in the authorization process, and it grants or denies access based on those attributes [20].

In the section 2 of the article, those security Requirements that must be followed in the field of access control are explained and there is a conceptual model of a collection of Requirements. Then in the section 3, there is a comprehensive classification of all available technologies that meet the security Requirements; all of them will be explained. The section 4 belongs to the description of IBAC ‹RBAC ‹ABAC and RAdAC models, here the ABAC model's benefits and structure is explained; the ABAC model is more compatible with SOA. ABAC model's description becomes more complete in the section 5 of the article, when we present a comparison between ABAC and RBAC models using an example.

2. SECURITY REQUIREMENTS IN ACCESS CONTROL

Security is an important issue that must be well-defined in SOA environment so that it could be used in implementing the web services. In general and according to Source [6], security in the environment can be defined in this way: "the sum of all techniques, methods, procedures and activities employed to maintain an ideal state specified through a set of rules of what is authorized and what is not in a heterogeneous, decentralized, and inter-connected computing system".

Security objectives provide a categorization of the most basic security needs of an asset. Define in [6]:"a statement of intent to counter identified threats and/or satisfy identified organization security policies and assumptions". Security objectives are also called security properties, security aspects, security concerns or security requirements.

Based on our proposed conceptual model as shown in Figure1 , the total security requirements in the area of access control are divided into two general categories: functional aspects and non-functional aspects that the non-functional aspect of Security are within the functional aspects of security and covers them.

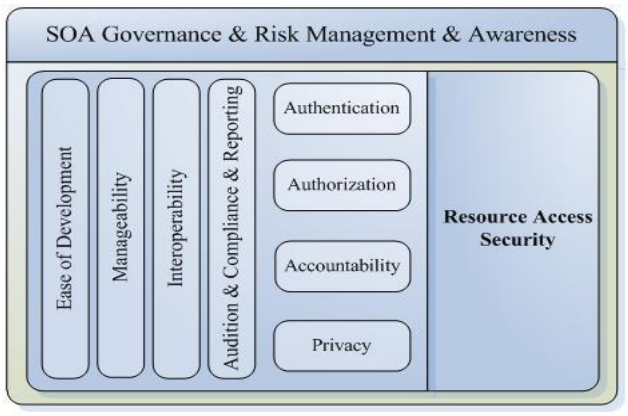


Figure1 : Security requirements conceptual model in access control

Functional aspects in the area include (Authentication, Authorization, Accountability and Privacy) and must seriously observance for more security in this area. Non-functional aspects include (Auditing & Compliance, Interoperability, Manageability and Ease of Development). Finally, both practical and impractical aspects of security in access control stand within the domain of SOA rule and training, awareness, and risk management.

2.1 Authentication

Authentication is the act of proofing that a claimed identity is true [7]. Therefore, authentication also involves identification. Identification involves the act of claiming an identity. Identification is usually the first step in the authentication and authorization process. There are three ways to authenticate an identity [8]:

- Knowledge based authentication: knowledge based revolves around the fact that there is some sort of knowledge that only the entity knows. This knowledge mostly has the form of a password. The idea is that is an entity knows the password it must be member of the group (a single user can or a group).
- Token based authentication: Something the entity has. An example of something that a person can have is a security badge, smartcard or a token. The idea is that if the entity has for instance a token, it must also have the claimed identity.
- Biometric authentication: Something the entity "is". This can be applied to biometric (usually the entity is a person); examples include fingerprints, iris scans and hand print.

2.1 Authorization

Authorization is a security concept where access to resources is allowed to only those who are permitted to use them. This is termed as access control and is usually determined by finding out if a person is a member of a particular privileged group [8].

Accountability

Accountability is to keep an account for provided determinant in communications taken place. Therefore people who share the communications can be responsible for their activities based on the account [13].

Privacy

Privacy is to keep the different informative parts of a message away from being seen by the entities not related to these parts. So the XML record must be parted and it also must have the ability to protect other parts despite service provider's interest [9].

Non-Functional Aspects of Security in Access Control

Security can be viewed as a three part process which involves protection, detection and response. Most of the security requirements discussed till now are protection schemes. We need a requirement which helps us to detect security vulnerabilities and respond with suitable measures which involves auditing.

A system should be configured to track messages between services and generate usage logs during specific periods of time. This audit serves as an important record of what has happened that can be used to investigate problems and diagnose potential security weaknesses.

Compliance is the state of being in accordance with established guidelines, specifications and legislation. It's not just archiving, but the ability to access information that is essential for achieving compliance. Every organization or enterprise has different service levels for different users, which drives information access, retrieval and disposition requirements. In essence, companies need to protect the right data longer, retrieve it as needed, and also discard it when it is obsolete [7].

Interoperability is unique concept for SOA and tells a different security solution should not violate the services are compatible [10]. Manageability means that, need security solutions to protect the various services. Good security architecture should be easy to manage [10].

Ease of Development is aspect of security solutions for all common. However the complexity of developing security solutions is higher. The possibility of adapting the architecture would be less relevant [10].

SOA Governance, Risk Management and Awareness

In fact it is the governance of tools for defining the organizational roles and allowing people to guiding and implementing the roles. There are different technical tools in the market to help automating the certain, defined aspects of governance process.

The other element in the layer is risk management. The risk could be defined as a combination of probability of an accident with its consequences. And risk Management, gradually and increasingly can find both positive and negative aspects of the risk. Since there would be many risks in SOA environment because of a wide range of different interactions and places, it is necessary to have a risk management ESP.

On the other hand, the issue of informing and educating the whole security requirements and concepts in SOA environment must be fulfilled so that the developers, users, managers and others are able to use it correctly when it is needed; this aim is achieved by awareness.

3. A COMPARISON OF AVAILABLE APPROACHES IN ACCESS CONTROL

Access management includes the access control and other related abilities such as identities management. Access management ensures that the resources are accessible only for users, programs, processes and/or permitted systems referring to access rules which are defined by policies and properties. One of the important aspects of SOA is that it has different users and sub-systems with the relationships and co- operations between them in doing the activities. In this architecture, recourses and services usually are shared by different users. The management of these entities, resources, access control levels, uncovers vulnerabilities and identifies potential threats and risks are posed as a challenge. This feature also can be seen in implementing this architecture, so the aspect of implementation should also be noted.

Based on our research in the field of SOA security, we've reached the conclusion that the SOA environment should be divided into (transport & network, message, interaction between services, services policies, access control and applications) domains, and Security should be respected in any field. Therefore we did polling among thirty Experts in computer and information technology, and we obtained useful results. One of the questions was the importance rate of security in the mentioned areas. Accordingly as shown in Figure 2, access control is the most important area and next, the message security is the second one with a little difference. These results further illustrate the importance of access control area and express the need for new approaches in this environment.

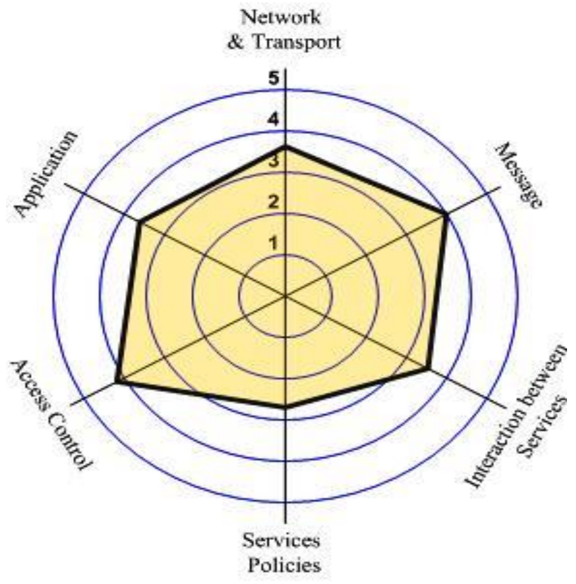


Figure 2 : Spider diagrams of the polling results in the rate of the importance of security areas

This section describe the authorization models most relevant to access management in a SOA, namely identity-based, role-based, attribute-based, and risk-adaptive access control.

3.1 Identity Based Access Control

Under this model, permissions to access a resource is directly associated with a subject’s identifier (e.g., a user name). Access to the resource is only granted when such an association exists. An example of IBAC is the use of Access Control Lists (ACL), commonly found in operation systems and network security services [20]. The concept of an ACL is very simple: each resource on a system to which access should be controlled, referred to as an object, has its own associated list of mappings between the set of entities requesting access to the resource and the set of actions that each entity can take on the resource.

Drawbacks of IBAC are: the number of identifiers in the ACL will increase and become difficult to maintain as more users request access, making this approach impossible to scale. The ACL for a particular file, process, or other resource must be checked every time the resource is accessed, and this can be an inefficient means of providing access control. Access control decisions are not based on any business function or characteristics of the user but solely on the identifiers, making it unsuitable for enterprise level use.

3.2 Role Based Access Control

The RBAC model restricts access to a resource based on the business function or role the subject is performing. The permissions to access a resource are then assigned to the appropriate role(s), rather than directly assigned to subject identifiers [16]. When a user changes jobs, some other user is allowed to take on that role. No ACL changes are needed. Of course, sometimes only a few of the user’s rights change. In that case, a new role needs to be introduced. Often the rights associated with a role depend on which user is acting in that role. In that case, too, a new role needs to be introduced [16]. The RBAC reference model is defined in terms of four model components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations [18]. Although RBAC may take slightly different forms, a common representation as defined in [17] is depicted in Figure 3.

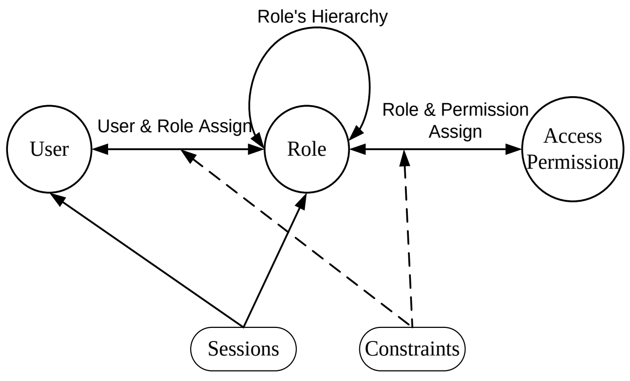


Figure3 : Role-based access control model

In most cases, Web service platforms will support designation and assignment of privileges to roles as part of their standard definition of user accounts and access control privileges. In worst cases, the administrator will have to create the necessary user groups, enroll the appropriate users, and assign them role-appropriate privileges. RBAC on a Web service platform should be implemented at a minimum for the administrator, developers, and any other privileged accounts that will be required for the Web service to operate. The Web service platform must be configured to enforce separation of roles (i.e., not allowing a user assigned to one role to perform functions exclusively assigned to another role). The privileges associated with each role should be assigned in a way that implements least privilege each role should be assigned only the minimum privileges needed to perform the functions required by the role [19].

Some RBAC limitations in SOA are: In RBAC you still have to manage every user account and bind these accounts to roles. Unknown accounts can only be linked to a default role, like guest or customer. Because the core RBAC model limits the abstraction of user function to roles only, it does not consider any other characteristics that a user may demonstrate. Further, RBAC generally doesn't take into account the characteristics of resources (other than their identifiers); nor does it capture any security relevant information of the environment.

3.3 Attribute Based Access Control

Policy-Based Access Control (PBAC), which is called Attribute-Based Access Control (ABAC) in the US Defense Department jargon, extends RBAC to a more general set of properties [22]. Unlike IBAC and RBAC, the ABAC model [20] can define permissions based on just about any security relevant characteristics, known as attributes. For access control purposes, we are concerned with three types of attributes:

- Subject Attributes (S). Associated with a subject that defines the identity and characteristics of that subject.
- Resource Attributes (R). Associated with a resource, such as a Web service, system function, or data.
- Environment Attributes (E). Describes the operational, technical, or situational environment or context in which the information access occurs.

In the most general form, a Policy Rule that decides on whether a subject s can access a resource r in a particular environment e, is a Boolean function of s, r, and e's attributes:

$$Rule\ X : can\_access(s, r, e) \leftarrow f(ATTR(s), ATTR(r), ATTR(e)).$$

ABAC clearly provides an advantage over traditional RBAC when extended into SOA environments, which can be extremely dynamic in nature. ABAC policy rules can be custom-defined with consideration for semantic context and are significantly more flexible than RBAC for fine-grained alterations or adjustments to a subject's access profile. ABAC also integrates seamlessly with XACML, which relies on policy-defined attributes to make access control decisions. One additional benefit to Web service implementations of ABAC lies in the nature of the loose definition of subjects. Because ABAC provides the flexibility to associate policy rules to any actor, it can be extended to Web service software agents as well [21].

One additional benefit to Web service implementations of ABAC lies in the nature of the loose definition of subjects. Because ABAC provides the flexibility to associate policy rules to any actor, it can be extended to Web service software agents as well. Figure 4, illustrates how an ABAC attribute authority (AA) can be integrated with a SAML framework. In this diagram, the AA generates attribute assertions, which contain all the attributes necessary for an access control decision based on an ABAC policy written in XACML. The PDP uses the attribute assertions, the authentication assertion, and the XACML policy to generate an authorization decision assertion [11].

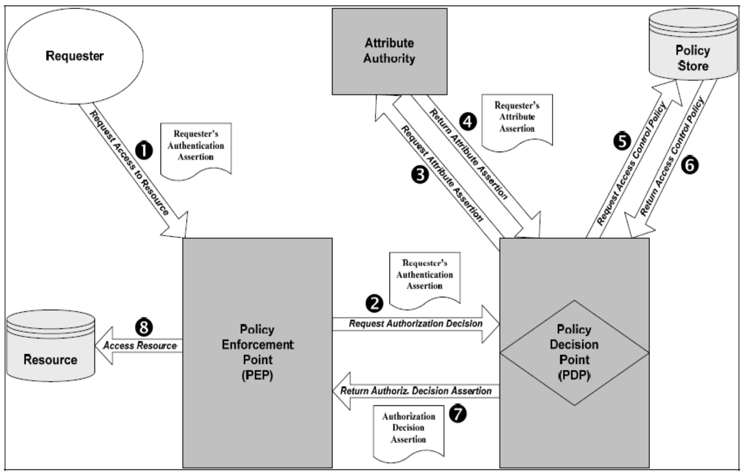

Requester
Attribute Authority
Policy Store
Request Access to Resource
Requester's Authentication Assertion
Request Attribute Assertion
Return Attribute Assertion
Requester's Attribute Assertion
Request Access Control Policy
Return Access Control Policy
Requester's Authentication Assertion
Request Authorization Decision
Resource
Access Resource
Policy Enforcement Point (PEP)
Policy Decision Point (PDP)
Return Authoriz. Decision Assertion
Authorization Decision Assertion
1
2
3
4
5
6
7
8

Figure 4 : Use of SAML and XACML in implementing ABAC

In Figure 4, the requester's authentication assertion is provided by the identity provider before accessing the resource. The following steps describe how SAML and XACML use the requester's attributes to determine whether access should be granted: (1) the requester attempts to access the resource and supply the authentication assertion. (2) The Policy Enforcement Point (PEP) sends a SAML authorization decision request to the PDP. (3) The PDP requests certain attribute assertions that are associated with the requester. (4) The AA returns the appropriate attribute assertions. (5) The PDP requests the XACML policy from the policy store. (6) The PDP receives the XACML policy. (7) After querying the XACML policy, the PDP sends an authorization decision assertion to the PEP. (8) Based on the authorization decision assertion, the PEP grants the requester access to the resource.

3.4 Risk Adaptive Access Control

Risk adaptive access control (RAdAC) [23] is another variation access control methods. As opposed to IBAC, RBAC and ABAC, however, RAdAC makes access control decisions on the basis of a relative risk profile of the subject and not necessarily strictly on the basis of a predefined policy rule.

Figure 5 illustrates the logical process governing RAdAC, which uses a combination of a measured level of risk the subject poses and an assessment of operational need as the primary attributes by which the subject's access rights are determined. As a policy-driven mechanism, RAdAC is ostensibly an abstraction of ABAC. Unlike ABAC, however, a RAdAC framework requires associations with sources that are able to provide real-time, situation aware information upon which risk can be assessed with each authentication request.

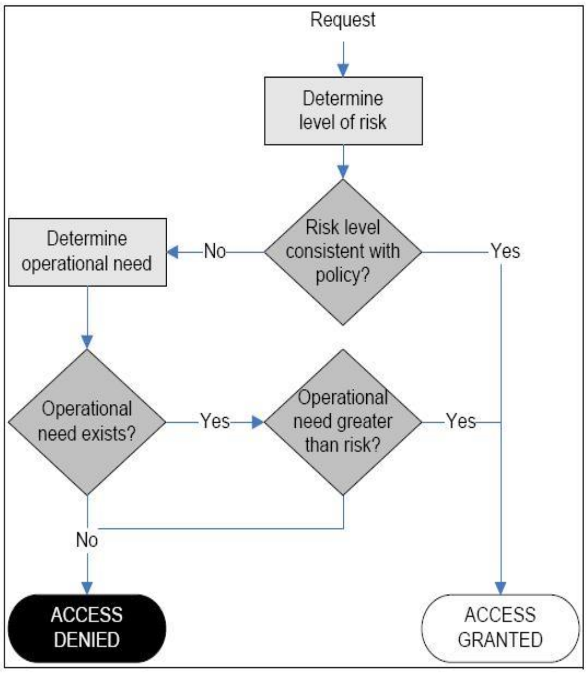

Request
Determine level of risk
Risk level consistent with policy?
No
Yes
Determine operational need
Operational need exists?
Yes
Operational need greater than risk?
Yes
No
ACCESS DENIED
ACCESS GRANTED

Figure 5 : RAdAC decision tree

4. COMPARE ABAC VS. RBAC WITH EXAMPLE

Since many access control solutions today are role based, in this section we compare ABAC with the latest RBAC approaches to illustrate the advantages of this new model. We use an example that involves a slightly more complex access control scenario to show how RBAC and ABAC attack the problem differently [20].

In example, an Online Entertainment Store streams movies to users for a flat monthly fee. The store needs to enforce an access control policy that is based on the users' age and the movie content ratings. The movie ratings and the corresponding access control policy is listed in table 1:

Table 1: movie rating and user allowed

Movie Rating	User Allowed
R	Age 21 or older
PG-13	Age 13 or older
G	Everyone

In the standard RBAC model, where only user roles are considered, there would be three pre-defined roles created for users, Adult, Juvenile, and Child. Each user of the store would be assigned to one of the three roles, possibly during registration. There would be three permissions created, namely, Can view R rated movies, Can view PG-13 rated movies and can view G rated movies, each represent the permission to view movies with the respective rating. The Adult role gets assigned with all three permissions, where as the Juvenile role gets Can view PG-13 rated movies and Can view G rated movies permissions, and the Child role gets the Can view G rated movies permission only.
Both the user-to-role assignment and the permission-to-role assignment are manual administrative tasks. In comparison, the ABAC approach in this scenario has no need to explicitly define roles. Instead, whether a user u can access or view a movie m (in a security environment e which is ignored here) would be resolved by evaluating a policy rule such as the following:

$$
\begin{aligned}
&R1{:}\ can\_access(u,m,e) \leftarrow \\
&\quad (Age(u) \geq 21 \wedge Rating(m) \in \{R, PG-13, G\}) \quad \vee \\
&\quad (21 > Age(u) \geq 13 \wedge Rating(m) \in \{PG-13, G\}) \quad \vee \\
&\quad (Age(u) < 13 \wedge Rating(m) \in \{G\})
\end{aligned}
$$

Where Age and Rating are the subject attribute and the resource attribute, respectively. The advantage of the ABAC model shown here is that it eliminates the definition and management of static roles, hence also eliminates the need for the administrative tasks for user-to-role assignment and permission-to-role assignment.

Finer-grained access control policies often involve multiple subjects and object attributes. In such cases ABAC becomes more manageable and scalable than RBAC. To illustrate this, we extend the example slightly: suppose movies are also categorized either as New Release or as Old Release based on release dates, and users are further classified as Premium User and Regular User based on the membership fee paid. Suppose the store would like to enforce a policy such that only premium users can view new releases.

In RBAC, the roles and permissions that need to be created have both doubled:

Table 2 : roles and permissions in RBAC

Standard RBAC Roles
Adult premium user role
Adult regular user role
Juvenile premium user role
Juvenile regular user role
Child premium user role
Child regular user role

Standard RBAC Permissions
Can view R rated new release
Can view R rated old release
Can view PG-13 rated new release
Can view PG-13 rated old release
Can view G rated new release
Can view G rated old release

In general, if there are K subject attributes and M resource attributes, and if for each attribute, Range() denotes the range of possible values it can take, then the respective number of roles and permissions need to be created would be:

$$\prod_{1}^{k} Range(SA_k) and \prod_{1}^{M} Range(RA_m)$$

Therefore, as the policy becomes finer-grained and more attributes are involved, the number of roles and permissions will grow exponentially, making the user-to-role assignment and permission-to-role assignment tasks prohibitively expensive.

In the ABAC model, the previous policy rule R3 still applies; we only need to add a second rule R4, and then combine the two conjunctively:

R4: can_access(u, m, e) ← (MembershipType(u) = ‘Premium’) ∨ (MembershipType(u) = ‘Regular’ ∧MovieType(m) = ‘OldRelease’)

R5: can_access(u, m, e) ← R3 ∧ R4

So far we haven’t addressed environment attributes. The RBAC model does not address environment attributes explicitly. For example, it would be even more awkward to implement a policy that states “Regular users are allowed to view new releases in promotional periods.” By contrast, in an ABAC model, this policy can be implemented simply by adding conjunctive sub-clauses to check to see if today’s date, an environment attribute, falls in a promotional period.

5. PROPOSED ENVIRONMENT FOR ACCESS CONTROLLING

In an access control model for Web Service, SAML makes a role to protect, transport, and request XACML instances and XACML defines the information exchanging between each entity such as request, response and policy. This model can be applied to access control between Portal and Web Services.

XACML is a powerful standard language that defines schemas for authorization policies and for authorization decision requests/responses and specifies how to evaluate policies, but confines its scope to the language elements used directly by the PDP and does not specify protocols or transport mechanisms. XACML also does not define how to implement PEP, PAP, PIP, context handler, or Repository. For example, the Policy Enforcement Point (PEP) sends the request to the Policy Decision Point (PDP) with the "Request" element defined in the XACML context. But XACML files can serve as a standard format for exchanging information between these entities when combined with other standards.

SAML is one standard suitable for providing the assertion and protocol mechanisms and specifies schemas for carrying the security and authorization related information and have the bindings to basic transportation mechanisms. Therefore, OASIS publishes a SAML profile for the XACML to carry the XACML messages between the XACML actors. This profile defines the usage of SAML 2.0 to protect, store, transport, request and respond with XACML instances and other information. The proposed architecture in controlling access to protected resources is shown in Figure 6.

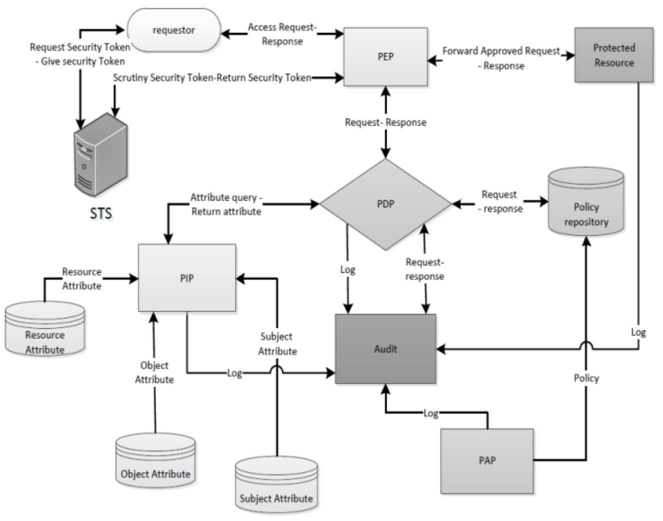


Figure 6 : The proposed architecture for access control

5.1 Authentication in proposed model

Authentication in service oriented architecture model, introduces some challenges. When a service is called from different ways and by users whose are in different organizations, how should the authentication process be done?

WS-Security, WS-Trust, and WS-Security Policy provide a basic model for federation between Identity Providers and Relying Parties. These specifications define mechanisms for codifying claims (assertions) about a requester as security tokens which can be used to protect and authorize web services requests in accordance with policy. WS-Federation extends this foundation by describing how the claim transformation model inherent in security token exchanges can enable richer trust relationships and advanced federation of services.

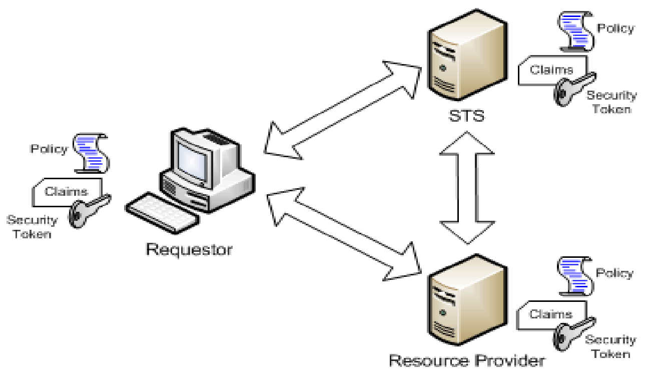


Figure 7 : Security Token Service (STS) Model

Figure 7, illustrates a view of the Security Token Service (STS) model defined by WS-Trust [12]. Each arrow represents a possible communication path between the participants. Each participant has its own policies which combine to determine the security tokens and associated claims required to communicate along a particular path. From the Requestor's perspective the communication flow starts with the identification of a web service, or Resource Provider, that the requestor wishes to access. The requestor queries the Resource Provider for its policies to determine security requirements. Using WS-Security Policy expressions the Requestor can check its own capabilities to determine if requestor has a security token that meets the requirements to access the Resource Provider. If the requestor does not have an acceptable token it might be able to request one from an appropriate STS which can also be identified in the Resource Provider's policy. Each STS has its own associated policy and being a web service, the Requestor can query the STS to determine the security requirements for requesting a particular type of token for use.

The WS-Trust specification defines the Authentication Type parameter to indicate a type of authentication that is required (or performed) or an assurance level with respect to a particular security token request. However, no particular types are defined or required. To facilitate interoperability WS-Federation has identified and defined a set of Universal Resource Identifiers (URIs) for specifying the common authentication types and assurance levels.

5.2 Use Case Diagram for the Proposed Architecture

Since this article focuses on access control security in web service environment, we review the possible operations in the access control domain. The main activities in the field are shown in the use case diagram of Figure 8.

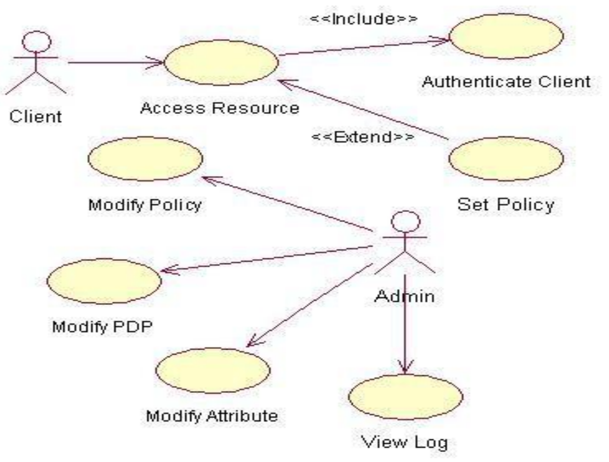


Figure 8 : Use Case diagram for proposed Architecture

In the diagram it is shown that a user can request an access to a protected resource. The request is accepted after user identification and if necessary after applying certain policies on the resource requested; the details of the process are explained in next part. On the other hand, the permission to change the policies in decision points, specifications and features intended for entities, and also the ability to view log has been provided for management.

5.3 Sequence Diagram for the Protected Resource

In order to have a better understanding of the work done when there is a request for a supported resource, we see the sequence diagram for the proposed architecture in Figure 9.

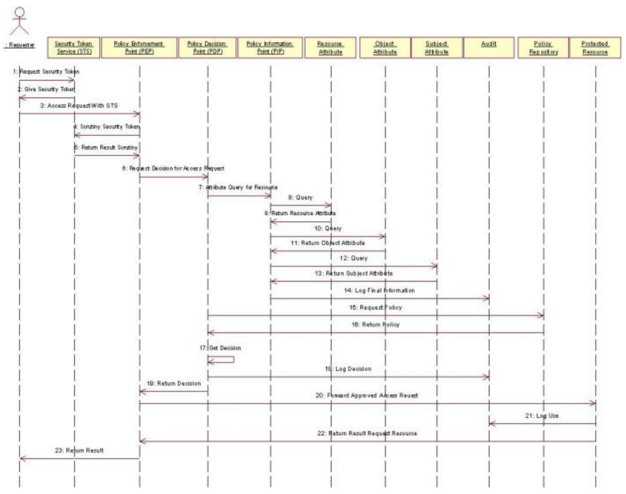
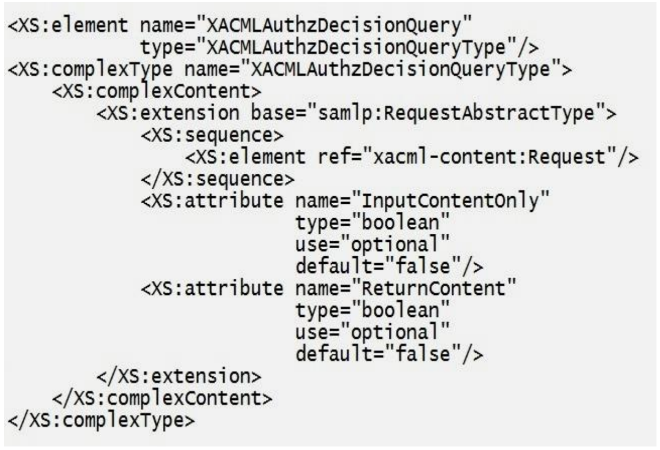

: Requester
Security Token Service (STS)
Policy Enforcement Point (PEP)
Policy Decision Point (PDP)
Policy Information Point (PIP)
Resource Attribute
Object Attribute
Subject Attribute
Audit
Policy Repository
Protected Resource
1: Request Security Token
2: Give Security Token
3: Access Request With STS
4: Scrutiny Security Token
5: Return Result Scrutiny
6: Request Decision for Access Request
7: Attribute Query for Resource
8: Query
9: Return Resource Attribute
10: Query
11: Return Object Attribute
12: Query
13: Return Subject Attribute
14: Log Final Information
15: Request Policy
16: Return Policy
17: Get Decision
18: Log Decision
19: Return Decision
20: Forward Approved Access Reuest
21: Log Use
22: Return Result Request Resource
23: Return Result

Figure 9 : The sequence diagram for using a protected resource

To complete the descriptions, below is a set of stages for a relationship between a web service program and a specific resource.

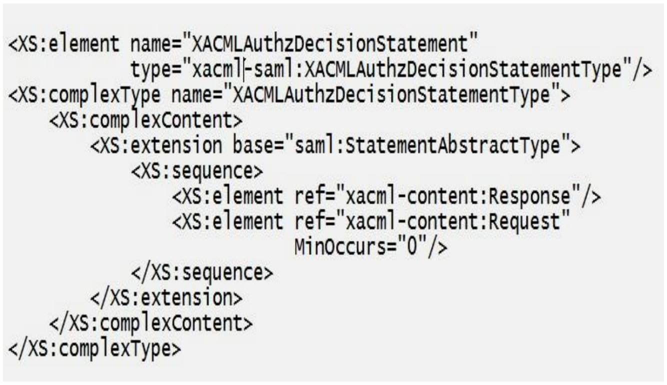
1) First, a user is connected to a web service program and requests the security token from STS for use of a protected (resource) service. STS gives the security token to user.
2) A SOAP message is generated and embeds a SAML authentication assertion in the SOAP header. These provisions include the user's security token that is ultimately sent for the PEP unit.
3) The handler on the Web service side acts as the SAML PEP. It intercepts the received SOAP message and requests the validation of the security token. Then it determines the assigned role of the requester as subject through identity management module, the operation name extracted from the SOAP body as action, and the name of Web service as resource. With these required factors, SAML PEP generates a XACMLAuthzDecisionQuery (request for decisions based on XACML) and sends it to SAML PDP. Structure of this request in the form of labels is shown in Figure 10.
4) SAML PDP is the point to actually make access decision. It retrieves the attributes required from attribute repository, such as LDAP or database. It retrieves the policies from policy repository, such as file system or a database, like dbXML, a native XML database to store XML. If the history of actions and attributes needs for some policies they will be retrieved from audit unit and then evaluates whether the request is granted. The results of every stage are saved in Review and the Log unit. Finally it constructs a XACMLAuthzDecisionStatement as a SAML response containing a XACML response context and sends it back to SAML PEP in response to the XACMLAuthzDecisionQuery. This step achieves the authorization of the user on the Web service side. The label for the answer is also shown in Figure 11. Also the decisions adopted are saved in the log unit.



```
<XS:element name="XACMLAuthzDecisionQuery"
            type="XACMLAuthzDecisionQueryType"/>
<XS:complexType name="XACMLAuthzDecisionQueryType">
    <XS:complexContent>
        <XS:extension base="samlp:RequestAbstractType">
            <XS:sequence>
                <XS:element ref="xacml-content:Request"/>
            </XS:sequence>
            <XS:attribute name="InputContentOnly"
                          type="boolean"
                          use="optional"
                          default="false"/>
            <XS:attribute name="ReturnContent"
                          type="boolean"
                          use="optional"
                          default="false"/>
        </XS:extension>
    </XS:complexContent>
</XS:complexType>
```

Figure 10 : Needed labels for the request XACMLAuthzDecisionQuery [15]

5) The PEP interprets the received answer and extracts the adopted decisions. If the result is PERMIT, the requester has the privilege to invoke Web service (resource); in other case the right to use the service is removed from user's program and it is not obtained.



```
<XS:element name="XACMLAuthzDecisionStatement"
          type="xacml-saml:XACMLAuthzDecisionStatementType"/>
<XS:complexType name="XACMLAuthzDecisionStatementType">
    <XS:complexContent>
        <XS:extension base="saml:StatementAbstractType">
            <XS:sequence>
                <XS:element ref="xacml-content:Response"/>
                <XS:element ref="xacml-content:Request"
                          MinOccurs="0"/>
            </XS:sequence>
        </XS:extension>
    </XS:complexContent>
</XS:complexType>
```

Figure 11 : Needed labels for the response XACMLAuthzDecisionStatement [15]

6) Finally, after calling the service and logging it, the result is sent to the applicant's program.

6. THE PROPOSED ARCHITECTURE'S BENEFITS

The proposed architecture and its benefits could be viewed from different points. Below is a set of some of its benefits.

- The architecture can extend and we can add other storages to it for adopting more accurate decisions and applying more policies. For example, the environment specifications' storage (including features of operations, techniques, environments and suitable times for using the service), resource specifications storage (including the features of resources and data) and so on.
- The architecture is based on ABAC approach. If new rules in accessing the resources are necessary, they can be defined dynamically, and then added to the approach.
- Standard languages with the ability to progress have been used in the approach for data transferring and stating policies. These languages support different kinds of data, functions and more that makes the implementation easy.
- Given to the architecture, access controlling could be done in any environment and a written policy can be used in many applications (applied programs).
- Stating the policies in the approach has a distributed structure. Certain people and groups as manager from different places can change the policies and specifications.
- Since the approach is provided with review and log service, all of the operations, decisions and returned results from the storages which are related to the accepted application, are saved so that they can be used for finding attacks, vulnerabilities and reports of stored records.
- Audit unit in this model can be used as history part of attributes and actions that happens; and PDP can queries from this part the information that it needs for making decision in future requests.

7. DISCUSSIONS

The article is on reviewing the security in the field of access control in the SOA environment. To obtain this purpose, security requirements, elements and approaches followed by a conceptual model and the available approaches and standards in the field are explained. Further, there is to present a framework for establishing security in access control field in the web service environment. For this purpose, we explained the needed elements in accesses' controlling including XACML and SAML languages followed by needed storages. We followed the article with offering architecture in controlling the access to protected resources based on ABAC model using a combination of two standard XACML and SAML languages; we also explained the process stages and the model's properties and benefits.

Acknowledgements

This Study Was Supported by Islamic Azad University, Borujerd Branch, Iran. The authors would like to acknowledge staffs of University.

REFERENCES

[1] T. Erl, "SOA: Principles of Service Design, " Prentice Hall/Pearson PTR, 2007.

[2] J. Wang, A. Yu, X. Zhang and L. Qu, "A Dynamic Data Integration Model Based on SOA, " In: 2009 ISECS International Colloquium on Computing, Communication, Control, and Management, pp. 196-199. IEEE, 2009.

[3] T. Parveen and S. Tilley, "A Research Agenda for Testing SOA-Based Systems," In: SysCon 2008-IEEE International Systems Conference, Montreal, Canada, April 2008.

[4] M. P.Papazoglou and W. Van Den Heuvel, "Service oriented architectures: approaches, technologies and research issues," pp. 389-415, Springer-Verlag, 2007.

[5] J. Eckert, M. Bachhuber, A. Miede, A. Pasageorgiou and R. Steinmetz, "Service-oriented Architectures in the German Banking Industry-A Multi-Participant Case Study," In: 4th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2010), 2010.

[6] M. Hafner and R. Breu, "Security Engineering for Service-Oriented Architectures, " Springer, 2009.

[7] V. Jonnaganti, “An Integrated Security Model for the Management of SOA- Improving the attractiveness of SOA Environments through a strong Architectural Integrity, ” Master Thesis, University of Gothenburg Department of Applied Information Technology Gothenburg, Sweden, 2009.

[8] J. Fiere, “SOA Security, ” Master Thesis, Faculty of Science Vrije Universiteit Amsterdam, 2007.

[9] D. Jana, A. Chaudhuri and B. Bhaumik, “Privacy and Anonymity Protection in Computational Grid Services, ” International Journal of Computer Science and Applications, Vol, 6, No, 1, pp. 98-107, 2009.

[10] R. Kanneganti and P. A.Chodavarapu, “SOA Security, ” Manning, 2008.

[11] A.Esfandi and M.Sabbari, "Study of Access Control Issue in Web Services" International Journal of Computer Applications, 2012, pp. 11-16.

[12] http://msdn.microsoft.com/en-us/library/bb498017.aspx.

[13] J. Janssen, “Identity management within an organization, ” Master Thesis, Radbound University Nijmegen, 2008.

[14] A.Esfandi, "Efficient and Secure Web Services by using Multi Agents." International Journal of Computer Applications, 2012, pp. 38-43.

[15] A. Anderson and H. Lochhart, “SAML 2.0 profile of XACML,” OASIS Standard, 2005.

[16] R. S.Sandhu and et al, “Role-Based Access Control Models,” IEEE Computer, pp. 38-47, 1996.

[17] J. WU and C. XI, “The Study on Service Oriented Access Control Model,” Second International Conference on Information and Computing Science, IEEE, 2009.

[18] R. Kuhn, “Role Based Access Control,” American National Standards Institute, 2003.

[19] D. Rolls, “Establishing an operational context for shared role-based access control systems,” White Paper, SailPoint Technologies, Jun. 2008.

[20] E. Yuan and J. Tong, “Attributed Based Access Control (ABAC) for Web Services, ” IEEE International Conference on Web Services (ICWS’05), 2005.

[21] J. Tong, “Attribute based access control: a new access control approach for service oriented architectures,” Workshop on New Challenges for Access Control, Ottawa, Canada, Apr. 2005.

[22] A. H. Karp and J. Li, “Solving the Transitive Access Problem for the Services Oriented Architecture,” IEEE International Conference on Availability, Reliability and Security, DOI 10.1109/ARES, 2010.

[23] P. C. Cheng, P. Rohatgi, and C. Keser, “Fuzzy MLS: an experiment on quantified risk–adaptive access control,” In 2007 Proc. IEEE Symposium on Security and Privacy, pp. 222-230.