IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-3, March 2019 E-ISSN: 2347-2693

An XML Based Framework For ABAC As A Service Based On Policy Machine Architecture

Vibha Bhardwaj[1*], Sushil Sharma[2]

[1,2]Department of Computer Science and Engineering, Institute of Technology and Management, AKTU, Aligarh, India

[*]*Corresponding Author: bharadwajvibha.vb@gmail.com, Tel.: +91-9068908199*

DOI: https://doi.org/10.26438/ijcse/v7i3.461469 | Available online at: www.ijcseonline.org

Accepted: 05/Mar/2019, Published: 31/Mar/2019

***Abstract*—** The cloud based systems offer Software as a Service (SaaS). This provides users a standard, robust, scalable & affordable software which they can access anytime from anywhere. The whole software may be composed of many small services which can be provided by different collaborating cloud service providers. The Service represents a software component that has the potential of reuse. In secured systems, Access Control Mechanism is very frequently used to restrict information flow to unauthorized users. Access Control Mechanism can be provided as a service so that it can be integrated with Software as a Service application. An Attribute Based Access Control (ABAC) mechanism is fine-grained, dynamic & scalable method to control access of the resources. A comprehensive policy can be deployed to specify access control rules. Policy Machine architecture can be used for policy specification and enforcement. Here we present an XML based framework to provide ABAC mechanism as a service based on Policy Machine architecture.

***Keywords*—**Access Control, Attribute Based Access Control, Service, Service Oriented Architecture, Policy Machine, XML

I. INTRODUCTION

Nowadays in this connected world through the Internet, people are performing their computing activities over the remote machines. Such machines are called servers which provide us services. A Service represents a business operation that has the potential of reuse. In today's scenario, the whole software is available as a service which in turn is an aggregation of small standalone services. This represents Service Oriented Architecture of software development where the basic building block of software is a service. People are using third-party services for their computing need. They share a lot of information while using these services. Due to the security and economic objectives, there is a need that the shared information should be available to the authorized users only. This follows the confidentiality aspect of system security. The applications use access control mechanisms to allow only authorized users to access the system. The Identity Based Access Control (IBAC) and Role Based Access Control (RBAC) mechanism typically used in Operating Systems and DBMS are two more popular forms of access control mechanism. Nowadays Attribute Based Access Control (ABAC) mechanism is gaining popularity. It is fine-grained, dynamic and scalable form of the access control mechanism which can adapt to real-world scenarios taking environment conditions into consideration. Instead of User-Permission or Role-Permission kind of relationship/rules, ABAC works on the security policy. The security policy specifies what is allowed and what is not allowed. A security policy is translated into an access control policy which is used by an access control mechanism. The Policy Machine framework can be used for policy specification and enforcement in ABAC mechanism. The enterprise applications provide diverse functionalities which are accessed by users located at separate geographies. In enterprise systems security policy is centrally defined and this centralized policy is adapted into local security policy. Therefore in order to maintain centralized control and usage by diverse functional application, there is a need to provide access control mechanism as a service so it can be integrated with software as a service application.

In this paper, we will present an XML based framework for ABAC as a Service based on Policy Machine Architecture. This framework will provide ABAC as a Service that is specified in terms of the combined construct of ABAC and Policy Machine Architecture. It standardizes interfaces for ABAC as a Service. It gives an XML based security policy specification which is based on ABAC and Policy Machine Architecture.

The rest of the paper is organized as follows, Section I contains the introduction of an XML based framework for ABAC as a Service based on policy machine architecture, Section II contains the related work in the field of access control, service and policy specification, Section III contain the architectural overview of ABAC as a Service, Section IV explain the policy specification and access control mechanism with flow chart, section V describes results and discussion in the form of an example application – Secured Image Server which uses ABAC as a Service, Section VI concludes research work with future directions.

II. RELATED WORK

NIST defines ABAC as ”An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environmental conditions, and a set of policies that are specified in terms of those attributes and conditions” [1]. So the ABAC system components are Subject, Object, Attributes, Operation, Policy and Environment conditions. ABAC is a logical access control model as it takes decisions dynamically by evaluating attributes, rules or policies and environmental conditions each time a request is made. The major components of enterprise ABAC are Enterprise ABAC Policy, Attribute Management, and Access Control Mechanism. The Access Control Mechanism of Enterprise ABAC contains functional points such as Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP) and Policy Administration Point (PAP). The Example of ACM functional points of Enterprise ABAC is given in [1].

The Policy Machine is a novel architecture and framework for access control policy specification and enforcement [2]. The main advantage of the PM architecture is that it reduces the amount of code that needs to be trusted. The basic elements of PM are authorized users (U), processes (P), operations (Op) and Objects (O). The PM defines three Relations: Assignment which corresponds to privilege a user has, Prohibitions which express what a user or process is not allowed to do and Obligations which are expressed as pattern/response relation. In PM an operation on the object by a subject is granted if and only if the user has such privilege on the object and the user/process is explicitly not denied such capability.

The earlier access control models were HRU, Bell-LaPaduLa & Chinese Wall etc. These access control models were categorized as Discretionary Access Control (DAC) & Mandatory Access Control (MAC). Later Role Based Access Control (RBAC) was presented in 1992. It helps in reducing the administrative complexity and support review or revoking of permissions assigned to users. It also simulates the business model as the businesses have roles. Because of this it was adopted in a large number of organizations and become very popular. There has been the effort in adding attributes to RBAC so that merged scheme could take advantages of both. Various Combination strategies and options for integrating attributes with RBAC including RBAC-A dynamic roles, RBAC-A role-centric and RBAC-A attribute-centric [3]. On the same line, Temporal RBAC & location based RBAC system has evolved. RBAC could coexist with ABAC as a role can be treated as an attribute. RBAC require significant time for role engineering before its usages similarly ABAC require significant time in attribute engineering and their testing.

ABAC is also there from the last decade. It has been implemented by various vendors. The UCON model focuses on usage control where authorizations are based on the attributes of the involved components [4]. It is attribute based but, instead of focusing on core ABAC concepts, it deals with advanced access control features such as mutable attributes, continuous enforcement, obligations, and conditions. Damiani et al have proposed a framework for attribute based access control in open environments [5]. Bonatti et al have presented an approach to logically formulate service access and information disclosure constraints according to related entity attributes [6, 7]. A service negotiation framework for requesters and providers to gradually expose their attributes has been developed [8, 9, 10]. Wang et al propose a framework that models an attribute based access control system using logic programming with set constraints of computable set theory [11]. This work presents an approach that uses set theory for defining the policy. Yuan and Tong have presented ABAC from the aspects of authorization architecture and policy formulation [12]. It emphasizes on enforcement of decision rather than policy formulation. Isabel F. Cruz et al have presented a location-aware role and attribute based access control system [13]. RBAC has been extended in this system to provide a dynamic association of roles to the user. Based on the attributes of user and resource, the user to privileges and privileges to resource associations are defined in this system. The system enables/disables roles based on the outcome of the physical to logical location mapping function. An attribute based access control ABAC presented by Xin Jin et al in [14]. The paper also discusses a mapping of ABAC to DAC, MAC & RBAC.

A service is nothing but an extended form of inter-process communication between two processes that run on two different machines and possibly implemented in two different technologies. The services have evolved from the Remote Procedure Call (RPC) mechanism in Distributed Computed Environment (DCE). DCOM & CORBA are considered as first generation framework for service which has DCE/RPC as their foundation. The JAVA-RMI (Remote Method Invocation) & Microsoft’s DotNet frameworks are considered as 2nd generation framework for distributed object systems on which services are formed. The Interface

Definition Language (IDL) is an important part of the 2nd generation framework. IDL is used to define interfaces, input/output data types & etc. Thus IDL serves as the service contract between the two processes in distributed object systems. IDL serves as a precursor to Web Service Description Language (WSDL).

In the late 1990s, Dave Winer of UserLand Software developed XML-RPC, which can be considered as a mark to the birth of web services. XML-RPC uses HTTP to transport instead of proprietary technology as in JAVA-RMI. Thus XML-RPC is based on request/response pattern. It also uses XML marshaling/unmarshaling to convert language dependent objects thus providing language neutrality. It is similar to serializing/deserializing. XML-RPC laid the foundation of the Simple Object Access Protocol (SOAP). SOAP is an Object Access protocol which also work on HTTP & XML [15]. It governs format & processing rules for information exchange between the two cooperating processes. SOAP becomes part of the W3C Web Services standard 1.2 for developing complex web services. REpresentational State Transfer (REST) is another way of developing web service. It is more oriented towards HTTP than SOAP messages i.e. it uses HTTP instead of SOAP messages. The orchestration of complex web services from simple web service led to the birth of Service Oriented Architecture (SOA). SOA is a form of application development where each software component represents a service. With the advent of SOA, a lot of standardization & specification occur in the field of web services. Web Services messaging specification led to the development of SOAP over UDP, SOAP Message Transmission Optimization Mechanism (MTOM) specification for sending binary data efficiently to & from web service, WS-Notification specification for disseminating information to a set other web services thus realizing publish/subscribe relationship [16, 17]. For metadata exchange between web services WSDL 2.0, Universal Description Discovery & Integration (UDDI) & WS-Resource framework kind of specification are available [18, 19]. Nowadays we are moving towards cloud based Software as a Service (SaaS) application development & usage scenario. In SaaS scenario, complex software is provided as a service to the user for access over the Internet. Such software is composed of the simple services which are provided by a single or multiple providers in collaboration with each other. Amazon Web Service (AWS) & Microsoft Azure are the examples of cloud based web services. The cloud service providers employ restful web-services for load balancing and Elastic Cloud Compute (EC2) i.e. compute power on demand in cloud systems [20, 21]. Co-operation among cloud service providers require security & trust mechanism to be built among the cooperating providers. Automatic data sharing secured with cryptographic mechanism facilitate such cooperation [22]. D - Phase with Decoy technology present an approach to thwart unauthorized access in the cloud systems [23]. The development of Security Assertion Markup Language (SAML), eXtensible Access Control Markup Language (XACML), WS-Security, WS-Federation, WS-Trust & etc. have facilitated to provide software as a service by cooperating providers.

ABAC is relatively new in the access control mechanism. The two Standards that apply ABAC are eXtensible Access Control Markup Language (XACML) and ANSI/INCITS Next Generation Access Control (NGAC) reference implementation. Axiomatic is the leading provider of fined grained ABAC solutions. Axiomatic Policy Server 6.0 provide ABAC using XACML. XACML is a widely used standard for ABAC.

There are two ways for developing web services i.e. SOAP based or REST based web services. JAX-WS and JAX-RS are two Java based technologies for developing such web services. WS-ReliableMessaging, WS-Security, WS-Addressing, MTOM, WS-Policy, WS-SecureConversation, WS-SecurityPolicy, WS-Trust specifications are implemented in the Apache CXF and .NET framework for web service development. Nowadays computing, storage, networking, database, analytics, management, developer tools, tools for the Internet of Things, application services & etc. are available as standalone services or aggregated services in your application accessible through the cloud. Such services are provided by Amazon, Google, Microsoft, & Salesforce. The most popular services are Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3).

III. ARCHITECTURE

ABAC as a Service has been developed keeping of the view that many applications or sub-applications of an enterprise application will use it to provide access control based on policies of the organization. The ABAC as a service is based on web-based client-server architecture i.e. it is a web service. The architecture of ABAC as a service for an enterprise application is shown in figure 1. ABAC as a Service Application contains four main modules apart from PDP, PIP & PAP which are given as follows:

- User Provisioning Module: This module provides functionality for creating and managing the users of the application. The user attributes relate to the subject attributes in ABAC mechanism. So we gain subject attributes and their values from this module.

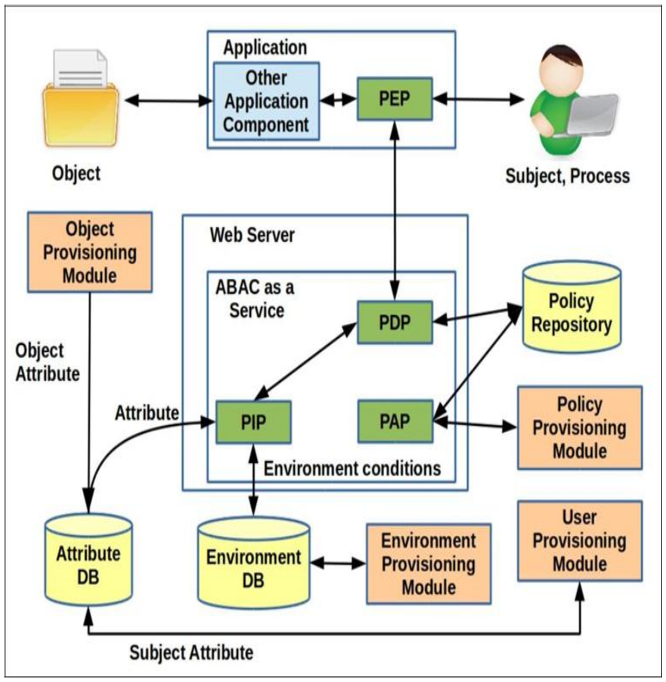


Figure 1. Architecture of ABAC as a Service for enterprise application

- Policy Provisioning Module: The module works in conjunction with PAP module. Together these modules provide interfaces for policy specification and administration. This module also holds the pointer to the currently applied policy. The policy is specified as per the combined construct of Policy Machine framework and ABAC defined in the section policy specification.
- Object Provisioning Module: This module creates & manages objects that exist in the system. So we get object attributes & their values from this module.
- Environment Provisioning Module: This module creates & manages properties that correspond to environmental conditions in the system. For example time, computer name, host name, IP address & past access history can be considered as environmental conditions. We get environment conditions from this module.

All the modules contain server and client-side components as ABAC is being provided as web-service. All the data resides on the server side.

IV. METHODOLOGY

This framework specifies the use of XML for policy specification, policy administration and input format specification for ABAC as a service. The XML has advantages as it is human readable, machine process-able and it is relatively easy to objectify entities using XML.

A. Policy Specification

This paper provides policy specification as a combined construct of the Policy Machine architecture & framework and ABAC standard. The PM defines a privilege relation as a triplet of the form (u, op, o) which specifies that a user u can perform operation op on object o. In ABAC a subject is identified by its attribute & attributes' value. Similarly, the object is identified by its attribute & attributes' value. In the combined construct u can be given as

$$u \Longrightarrow (S_a = S_{av})$$

Where S_a = attribute & S_{av} = attribute value for a subject S. Similarly, o can be given as

$$o \Longrightarrow (O_a = O_{av})$$

Where O_a = attribute & O_{av} = attribute value for an object O. Then a PM privilege can be represented as

$$\text{privilege} \Longrightarrow (u, op, o) \Longrightarrow (S_a = S_{av}, op, O_a = O_{av})$$

The attribute values can be numeric, alphanumeric, text or values from a set. The policy rules may be specified in a way that there is a need to perform the relational and logical operation for evaluating a rule. For example, a rule may be specified as any subject S which has age ≥ 25 can view object O with id = 5. Such a rule can be specified as

$$\text{privilege} \Longrightarrow (u, op, o) \Longrightarrow (S_{age} \geq 25, \text{view}, O_{id} = 5)$$

The relational operators considered in the framework are

$$\text{relational operator} \in \{>, \geq, <, \leq, =, \neq\}$$

Thus the number of attribute's values depend upon the definition of the rule. The rule may be defined in a way such that it considers attributes' values from a set, range or pattern. Thus the framework also considers IN, LIKE & BETWEEN operator for evaluating values from a set, range, and pattern. The IN operator considers the discrete values from a set. The LIKE operator works on a pattern and BETWEEN operator works on range values i.e. the value must be between [lower value, upper value]. So the operators that work on subject's attribute are

$$S_{op} \in \{>, \geq, <, \leq, =, \neq, \text{IN, LIKE, BETWEEN}\}$$

Where S_{op} is the operator that work on subject's attribute. Similarly, O_{op} represents operator that work on an object's attribute. O_{op} have the same operators as S_{op}. A privilege rule in a generic form can be specified as

$$\text{privilege} \Longrightarrow \left(\left(S_a\ S_{op}\ \{S_{av1}, \ldots\}\right), op, \left(O_a\ O_{op}\ \{O_{av1}, \ldots\}\right)\right)$$

The ABAC standard also considers environmental conditions. The rule may or may not contain environmental conditions as per its specification. A rule may contain one or more environmental conditions to evaluate. The privilege rule with environmental conditions can be given as

$$\text{privilege} \Longrightarrow \left(\left(S_a\ S_{op}\ \{S_{av1}, \ldots\}\right), op, \left(O_a\ O_{op}\ \{O_{av1}, \ldots\}\right), [\text{env\_conditions}]\right) \quad (1)$$

Where [env_conditions] are optional to the rule. The complex environmental condition may be formed by combining individual environment conditions using AND, OR operations. The complement of an environmental condition can be formed using NOT operation. The complex environmental conditions will be of the form given as follows:

$$\text{env\_conditions} \Longrightarrow ([\text{NOT}]\text{env\_cond}_1\ (\text{AND/OR})[\text{NOT}]\ \text{env\_cond}_2\ (\text{AND/OR})\ldots(\text{AND/OR})[\text{NOT}]\text{env\_cond}_n) \quad (2)$$

Where [NOT] is optional & (AND/OR) means either of the AND or OR. The framework considers time, the identity of requesting computer, location, and protocol of the request as the environmental conditions for policy specification. The identity of the computer could be IP based or MAC id based. The protocol considered in the framework is HTTP i.e. only the HTTP request are considered by the framework for policy specification & enforcement. The location considered by the framework is the location shared by the browser of the client. This location can be considered as a user's location and rules can be formed to grant/deny access based on the location of the user. The privilege rules with environmental conditions are more restrictive than privilege rules without environment conditions.

The PM also defines Prohibition relation which is of the form u_deny(u, ops, os) or p_deny(p, ops, os) where $u \in U$; $p \in P$; $ops \in 2^{op}$ & $os \in 2^{o}$. The prohibition relation states that a user u or process p is denied specified operations on the specified objects. This framework only considers the u_deny type of relation. So a prohibition relation in this framework is of the form:

$$\text{prohibition} \Longrightarrow \text{u\_deny}(u,\ ops,\ os)$$
$$\Longrightarrow \text{u\_deny}(u, \{op_1,\ldots op_n\}, \{O_1,\ldots,O_n\})$$

On combining the ABAC concepts with PM, the prohibition will be of the form:

$$\text{prohibition} \Longrightarrow \text{u\_deny}(S_a = S_{av}, \{op_1,\ldots op_n\}, \{O_{1a} = O_{1av},\ \ldots,\ O_{na} = O_{nav}\})$$

Similar to the privilege rule given in eq. (1), the prohibition rule in generic form can be given as follows:

$$\text{prohibition} \Longrightarrow \text{u\_deny}(\ (S_a\ S_{op}\ \{S_{av1},\ldots\}),\{op_1,\ldots op_n\},\ \{(O_{1a}\ O_{1op}\ \{O_{1av1},\ldots\}),\ldots,(O_{na}\ O_{nop}\ \{O_{nav1},\ldots\})\},\ [\text{env\_conditions}]) \quad (3)$$

The env_conditions in eq: (3) is of the form similar to given in the eq: (2). The prohibition rules with environmental conditions are less restrictive than the prohibition rules without environmental conditions. So the restrictiveness of the rules is ordered from privilege rules to prohibition rules where privilege rules without environment conditions being less restrictive to prohibition rules without environment conditions being most restrictive. This relation is shown in figure 2 given as follows:

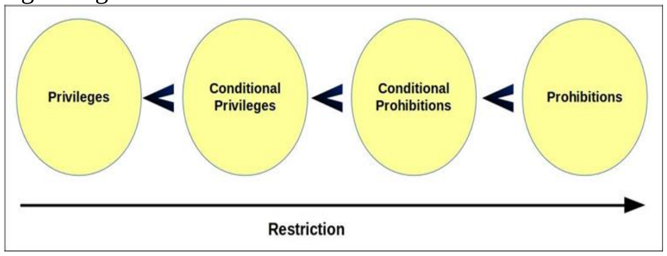
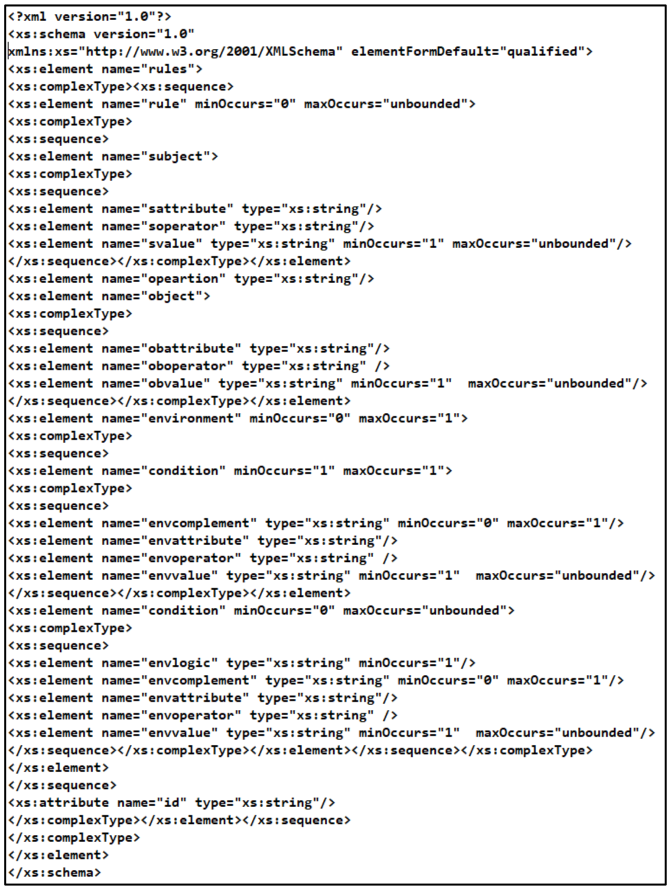


Figure 2. Restriction relation among policy rules

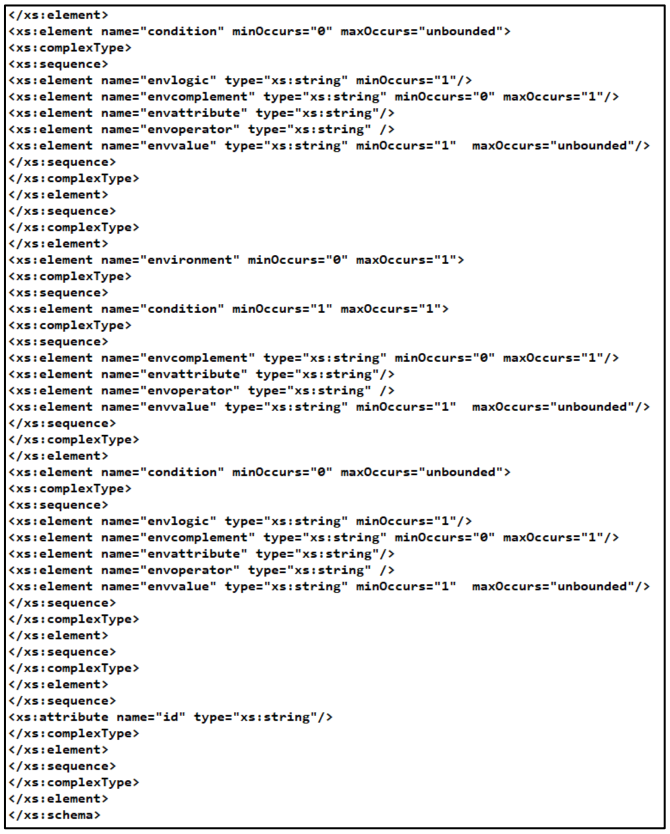
The policy specification is based on privilege and prohibition rules of the form specified in eq. (1) & (3). The framework uses XML specification for specifying rules of the policy. The framework defines XML Schema i.e. privilegeschema.xsd for policy specification that contains privilege rules as per eq. (1) specified in XML format. The rules are identified by their ids which help in accounting. A privilege XSD file containing schema is given below:



```
<?xml version="1.0"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="rules">
<xs:complexType><xs:sequence>
<xs:element name="rule" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="subject">
<xs:complexType>
<xs:sequence>
<xs:element name="sattribute" type="xs:string"/>
<xs:element name="soperator" type="xs:string"/>
<xs:element name="svalue" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence></xs:complexType></xs:element>
<xs:element name="opeartion" type="xs:string"/>
<xs:element name="object">
<xs:complexType>
<xs:sequence>
<xs:element name="obattribute" type="xs:string"/>
<xs:element name="oboperator" type="xs:string" />
<xs:element name="obvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence></xs:complexType></xs:element>
<xs:element name="environment" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="condition" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="envcomplement" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="envattribute" type="xs:string"/>
<xs:element name="envoperator" type="xs:string" />
<xs:element name="envvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence></xs:complexType></xs:element>
<xs:element name="condition" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="envlogic" type="xs:string" minOccurs="1"/>
<xs:element name="envcomplement" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="envattribute" type="xs:string"/>
<xs:element name="envoperator" type="xs:string" />
<xs:element name="envvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence></xs:complexType></xs:element></xs:sequence></xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" type="xs:string"/>
</xs:complexType></xs:element></xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Figure 3. Schema for privilege policy rules

The framework also defines XML Schema for policy specification part that contains prohibition rules. The prohibition schema defines standard according to which prohibition rules must be specified in XML format. A prohibition XSD file containing schema is given as follows:

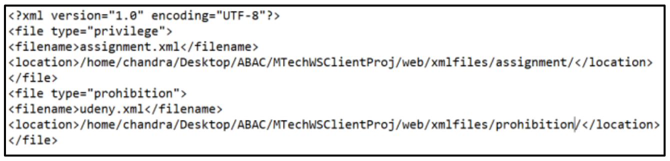


```
</xs:element>
<xs:element name="condition" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="envlogic" type="xs:string" minOccurs="1"/>
<xs:element name="envcomplement" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="envattribute" type="xs:string"/>
<xs:element name="envoperator" type="xs:string" />
<xs:element name="envvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="environment" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="condition" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="envcomplement" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="envattribute" type="xs:string"/>
<xs:element name="envoperator" type="xs:string" />
<xs:element name="envvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="condition" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="envlogic" type="xs:string" minOccurs="1"/>
<xs:element name="envcomplement" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="envattribute" type="xs:string"/>
<xs:element name="envoperator" type="xs:string" />
<xs:element name="envvalue" type="xs:string" minOccurs="1"  maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Figure 4. Schema for prohibition policy rules

B. Policy Administration

Policy specification and administration is provided by Policy provisioning module in conjunction with PAP. The module provides functionality for defining and storage of security policies i.e. creation, modification, deletion and saving of privilege & prohibition policy file by the administrator on the server. ABAC is dynamic. Keeping a view of this, the framework provides a facility to change the policy during runtime. The framework also provides functionality to link current policy files. This information is captured through an XML file as shown in figure 5. Policy Information Point (PIP) uses this information for extracting security policy details.

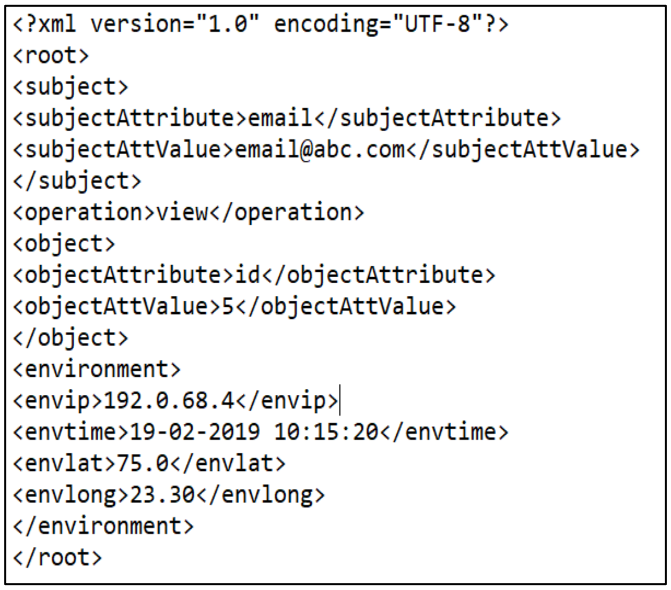


```
<?xml version="1.0" encoding="UTF-8"?>
<file type="privilege">
<filename>assignment.xml</filename>
<location>/home/chandra/Desktop/ABAC/MTechWSClientProj/web/xmlfiles/assignment/</location>
</file>
<file type="prohibition">
<filename>udeny.xml</filename>
<location>/home/chandra/Desktop/ABAC/MTechWSClientProj/web/xmlfiles/prohibition/</location>
</file>
```

Figure 5. Policy linking using XML

C. Input Specification

Since ABAC as a service is provided as a web-service by this framework, so there is a need to standardize the input to the service. The service needs to know about Subject, Object, intent (operation) and environmental conditions in which the request arise. The consumer of service provides all these details to the service in the format accepted by the service. This framework specifies the input as per the combined construct of ABAC & PM as defines by this framework. Figure 6 defines an XML format for specifying input of the service. The output of the service is very simple i.e. access is granted or denied.



```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<subject>
<subjectAttribute>email</subjectAttribute>
<subjectAttValue>email@abc.com</subjectAttValue>
</subject>
<operation>view</operation>
<object>
<objectAttribute>id</objectAttribute>
<objectAttValue>5</objectAttValue>
</object>
<environment>
<envip>192.0.68.4</envip>
<envtime>19-02-2019 10:15:20</envtime>
<envlat>75.0</envlat>
<envlong>23.30</envlong>
</environment>
</root>
```

Figure 6. ABAC as a service input specification using XML

D. ABAC Mechanism

The framework provides ABAC as a web-service. The ABAC mechanism is very straight forward. The ABAC as a service accepts the request to access an object by a subject. Since ABAC as a service is used by other applications, the other applications extract necessary information as given in figure 6 from the session and send it to ABAC mechanism as the HTTP request. The ABAC mechanism extract user, object, operation and environment information from the request and supplement this information by fetching relevant information such as subject's attributes, subject's attributes' values, object's attributes and object's attributes' values from the attribute database. ABAC mechanism also extracts environmental details. After this ABAC mechanism fetch rules from the security policy files. Then ABAC mechanism evaluates request against security rules specified in privilege and prohibition policy files. The access is granted if and only if there is a privilege rule which authorizes the access and there is no prohibition rule which denies the access. The flow chart of ABAC mechanism is shown in figure 7.

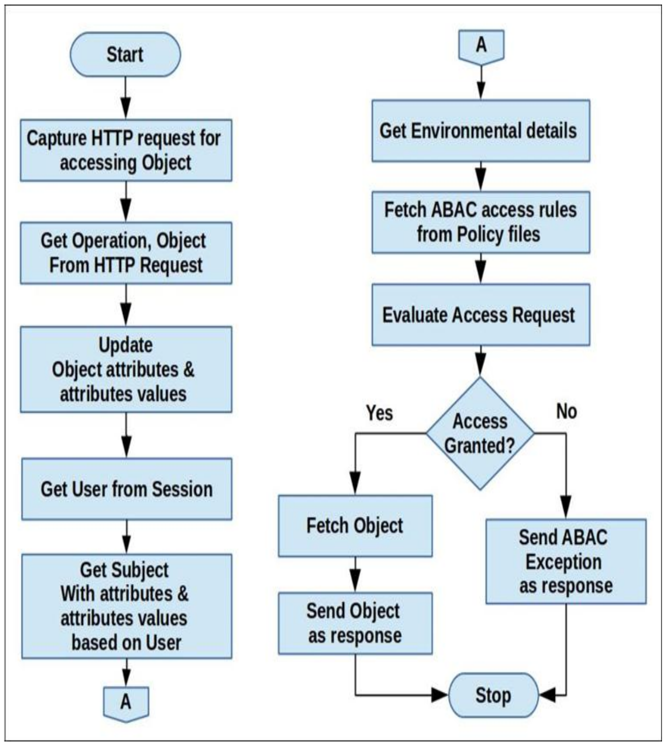
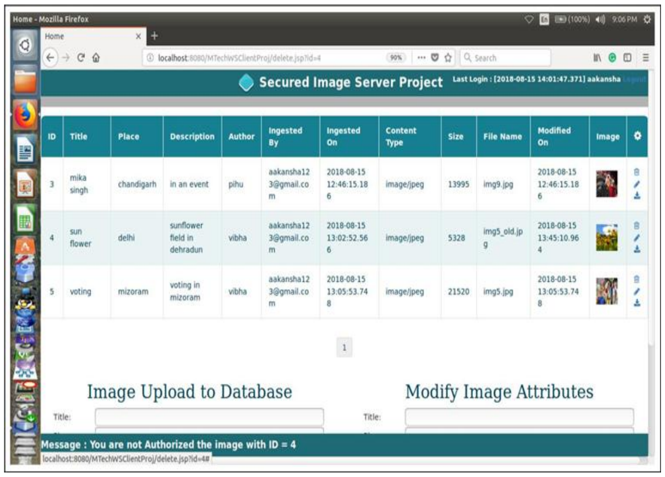

Start
Capture HTTP request for accessing Object
Get Operation, Object From HTTP Request
Update Object attributes & attributes values
Get User from Session
Get Subject With attributes & attributes values based on User
A
A
Get Environmental details
Fetch ABAC access rules from Policy files
Evaluate Access Request
Access Granted?
Yes
No
Fetch Object
Send Object as response
Send ABAC Exception as response
Stop

Figure 7. Flow chart showing ABAC mechanism

V. Results and Discussion

A. Secure Image Server (SIS) Application

The Secure Image Server application has been developed to demonstrate the use of ABAC as a service. The application is used for securely manage the images stored on the server. The application provides functionalities to view, upload, download and delete images from the server. The application demonstrates confidentiality aspect of the security i.e. application provides all these functionalities to the authorized user only using ABAC. ABAC as a service is governed by a security policy described in terms of privilege and prohibitions rules as specified in section 4. The application has two parts i.e. Secure Image Server application & ABAC as a service which has been developed as a web-service. The Secure Image Server application part uses ABAC as a service part to provide access control. The ABAC as a service has been developed using JAX-WS framework. The whole application is a web application i.e. both part are the web applications. The application is hosted on the application server i.e. GlassFish. It is based on the client-server model. The client/user access the secure image server application through web browser i.e. Mozilla Firefox. The user sends the request to access the object i.e. image in this case. The PEP point intercepts the request and forward it to the ABAC module. ABAC module evaluates the request and sends access control decision i.e. Grant/Reject decision to PEP. PEP performs the operation as per the access control decision. The screenshot of the application is given in the figure 8.


Secured Image Server Project Last Login : [2018-08-15 14:01:47.371] aakansha

ID	Title	Place	Description	Author	Ingested By	Ingested On	Content Type	Size	File Name	Modified On	Image
3	mika singh	chandigarh	in an event	pihu	aakansha123@gmail.com	2018-08-15 12:46:15.186	image/jpeg	13995	img9.jpg	2018-08-15 12:46:15.186	
4	sun flower	delhi	sunflower field in dehradun	vibha	aakansha123@gmail.com	2018-08-15 13:02:52.566	image/jpeg	5328	img5_old.jpg	2018-08-15 13:45:10.964	
5	voting	mizoram	voting in mizoram	vibha	aakansha123@gmail.com	2018-08-15 13:05:53.748	image/jpeg	21520	img5.jpg	2018-08-15 13:05:53.748	

Image Upload to Database

Title:

Modify Image Attributes

Title:

Message : You are not Authorized the image with ID = 4

localhost:8080/MTechWSClientProj/delete.jsp?id=4#

Figure 8. Secured Image Server application screenshot

B. Database

The database of Secured Image Server application has two main tables which capture the subject and object information. The Postgres database management system has been used to define SIS database. The User Table captures subject/user attributes & attribute values. The Image Table contains information about object/image attributes & attribute values. The attributes of the subject/object that need consideration depend upon the domain of application, for example, the image table contains attributes such as uploadedby, modifiedby & etc. that play important role in policy specification in SIS application. The policy may specify a rule such as a user X can modify image uploaded by user Y. Thus attribute specification is an important part of such application. The schema of both tables is given in Table 1 & Table 2 as follows:

Table 1. User Table

Column (Attributes)	Type	Constraints
username	character varying (15)	not Null
emailid	character varying (25)	not Null, PK
password	character varying (15)	not Null
age	numeric (3,0)	not Null
gender	character (1)	not Null
first name	character varying (15)	not Null
middle name	character varying (15)	
last name	character varying (15)	

Table 2. Image Table

Column (Attributes)	Type	Constraints
id	serial	not Null, PK
title	character varying (50)	not Null

Column (Attributes)	Type	Constraints
id	serial	not Null, PK
filename	character varying (50)	not Null
type	character varying (15)	not Null
size	bigint	not Null
author	character varying (50)	not Null
place	character varying (50)	
uploadedby	character varying (50)	not Null
uploadedon	timestamp without timezone	not Null
modifiedby	character varying (50)	
modifiedon	timestamp without timezone	
about	character varying	
imagedata	bytea	not Null

VI. CONCLUSION AND FUTURE SCOPE

Nowadays people access applications over the Internet which are provided as Software as a Service application to carry their business activities. The complex services are formed with the aggregation of simple services. Such cooperating services often store, process & share user data along with user's personal information. This user's data need to be protected from unauthorized access. This XML based framework describes a service based approach on the combined construct of ABAC & Policy Machine which provide a fine-grained, dynamic & scalable access control in web & service applications.

The applications associate with a large number of entities. This results in a number of attributes associated with the application. It becomes difficult to identify attributes that are relevant to make authorization decisions and coming out with policies from them. So a study can be carried out to identify probable attributes based on the domain of application and candidate policies can be given as templates/specification. The experience of this implementation can be carried forward to generate Optimized ABAC engine for SaaS applications. The optimization can be done on the basis of the domain of the application. The ABAC service based on policy machine architecture can be extended to build intrusion detection & prevention systems.

REFERENCES

[1] V.C. Hu et al., "*Guide to Attribute Based Access Control (ABAC) Definition and Considerations*", NIST Special Publication 800-162, **USA**, pp.**4-14**, **2014**.

[2] D. Ferraiolo, V. Atluri, S. Gavrila, "*The Policy Machine: a Novel Architecture and Framework for Access Control Policy Specification and Enforcement*", Journal of Systems Architecture, Vol.**57**, No.**4**, pp.**412-424**, **2011**.

[3] D.R. Kuhn, E.J. Coyne, T.R. Weil, "*Adding Attributes to Role Based Access Control*", IEEE Computer, Vol.**43**, No.**6**, pp.**79-81**, **2010**.

[4] J. Park, R. Sandhu, "*The UCONabc usage control model*", ACMTransactions on Information and System Security, Vol.**5**, No.**6**, pp.**128-174**, **2007**.

[5] E. Damiani ; S.D.C. di Vimercati ; P. Samarati, "*New paradigms for access control in open environments*", In The Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2005), Athens, **Greece**, pp.**540-545**, **2005**.

[6] P. Bonatti, P. Samarati, "*Regulating service access and information release on the web*", In The Proceedings of the 7th ACM conference on Computer and communications security (CCS 2000), Athens, **Greece**, pp.**134-143**, **2000**.

[7] P. Bonatti, P. Samarati, "*A uniform framework for regulating service access and information release on the web*", Journal of Computer Security, Vol.**10**, No.**3**, pp.**241-271**, **2002**.

[8] T. Yu, X. Ma, M. Winslett, "*Prunes: an efficient and complete strategy for automated trust negotiation over the internet*", In The Proceedings of the 7th ACM conference on Computer and communications security (CCS 2000), Athens, **Greece**, pp.**210-219**, **2000**.

[9] T. Yu, M. Winslett, K.E. Seamons, "*Interoperable strategies in automated trust negotiation*", In The Proceedings of the 8th ACM conference on Computer and communications security (CCS 2001), Philadelphia, **USA**, pp.**146-155**, **2001**.

[10] T. Yu, M. Winslett, K.E. Seamons, "*Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation*", Journal of ACM Transactions on Information and System Security (TISSEC), Vol.**6**, No.**1**, pp.**1-42**, **2003**.

[11] L. Wang, D. Wijesekera, S. Jajodia, "*A logic based framework for attribute based access control*", In The Proceedings of the 2004 ACM workshop on Formal methods in security engineering (FMSE 2004), Washington DC, **USA**, pp.**45-55**, **2004**.

[12] E. Yuan, J. Tong, "*Attributed based access control (ABAC) for web services*", In The Proceedings of the IEEE International Conference on Web Services (ICWS 2005), Washington DC, **USA**, pp.**561-569**, **2005**.

[13] I. F. Cruz, R. Gjomemo, P. Lin, M. Orsini, "*A location aware role and attribute based access control system*", In The Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS 2008), California, **USA**, pp.**527-528**, **2008**.

[14] X. Jin, R. Krishnan, R. Sandhu, "*A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC*", N. Cuppens-Boulahia et al. (Eds.): DBSec 2012, LNCS 7371, Frankfurt, **Germany**, pp.**41-55**, **2012**.

[15] M. Gudgin et al., "*SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*", W3C Recommendation, **USA**, pp. 1-10, 2007.

[16] M. Gudgin, N. Mendelsohn, M. Nottingham, H. Ruellan, "*SOAP Message Transmission Optimization Mechanism*", W3C Recommendation, **USA**, pp. 1-5, 2005.

[17] S. Graham, D. Hull, B. Murray, "*Web Services Base Notification 1.3*", OASIS Standard, **USA**, pp. 1-68, 2006.

[18] R. Chinnici, J. Moreau, A. Ryman, S. Weerawarana, "*Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*", W3C Recommendation, **USA**, pp. 1-17, 2007.

[19] R. Chinnici, H. Haas, A. A. Lewis, J. Moreau, D. Orchard, S. Weerawarana, "*Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts*", W3C Recommendation, **USA**, pp. 1-22, 2007.

[20] T. Agarwal, N. Sharma, "*Efficient Load Balancing Using Restful Web Services in Cloud Computing: A Review*", International Journal of Scientific Research in Computer Sciences and Engineering, Vol.**6,** No.**3**, pp.**67-70**, **2018**.

[21] A. A. Ekre, N. M. Nimbarte, S.V. Balamwar, "*An Empirical Proposition to Load Balancing Effectuate on AWS Hybrid Cloud*", International Journal of Scientific Research in Computer Sciences and Engineering, Vol.**6,** No.**4**, pp.**9-17**, **2018**.

[22] R. Bhavani, K. S. Suganya, D.Y. Priyanka, "*Autonomous PHR Sharing: A Patient Centric Scalable and Flexible e-Healthcare Framework*", International Journal of Scientific Research in Network Security and Communication, Vol.**6,** No.**2**, pp.**11-14**, **2018**.

[23] P. Devi, "*Attacks on Cloud Data: A Big Security Issue*", International Journal of Scientific Research in Network Security and Communication, Vol.**6,** No.**2**, pp.**15-18**, **2018**.

Authors Profile

Mrs. Vibha Bhardwaj pursed Bachelor of Technology from Mangalayatan University, Aligarh, India in 2011. She is pursuing Master of Technology from Institue of Technology & Management, Aligarh, an institute of AKTU University, Lucknow, India. Her main research work focuses on Access Control Systems in System Security.

Mr. Sushil Sharma pursed Master of Computer Application in 2005 and Master of Technology in 2012 from Uttar Pradesh Technical University of Lucknow, India. He is currently working as Dean in ITM, Aligarh since 2014. He is an Assistant Professor in the Department of Computer Science, ITM, Aligarh. He has published more than 4 research papers in reputed journals and conferences. His main research work focuses on Artificial Intelligence, System Security and Big Data Analytics. He has 14 years of teaching experience.