Interoperability among Access Control Models

By
Khalid Hafeez
2009-NUST-MS-PhD IT-16

Supervisor
Dr.Awais Shibli
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(March 2013)

Approval

It is certified that the contents and form of the thesis entitled "**Interoperability among Access Control Models**" submitted by **Khalid Hafeez** have been found satisfactory for the requirement of the degree.

Advisor: **Dr.Awais Shibli**
Signature: ____________________

Date: ____________________

Committee Member 1: **Dr. Zahid Anwar**

Signature: ____________________
Date: ____________________

Committee Member 2: **Dr. Abdul Ghafoor**

Signature: ____________________
Date: ____________________

Committee Member 3: **Qasim Rajpoot**

Signature: ____________________
Date: ____________________

I dedicate this thesis to my mother who have been so close to me that I found her with me when I needed. It is her unconditional love that motivates me to set higher targets.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Khalid Hafeez**
Signature: ________________

Acknowledgment

First of all, I am grateful to Allah almighty, WHO helped me in challenges that were faced while accomplishing this task. He is the One, gave me courage and sustainability whenever I was in need.
I also thank my Family for their support, specially my mother for her prayers and care at all times. My deepest gratitude goes to my research advisor Dr. Awais Shibli for his support and guidance, for his funding of precious advices during thesis work and unmatched concern towards my overall well-being. I simply could not think of a more deep-sighted, knowledgeable and practical supervisor, who also understands the psychology of his pupils and introduces to them the finer points of the subject in the easiest possible manner.
I am full of appreciation to the respected committee members Mr. Qasim Rajpoot, Dr. Zahid Anwar and Dr. Abdul Ghafoor for their cooperation and efforts during my thesis.
My SEECS fellows and colleagues duly deserve admiration for backing me up and assisting me in many little ways yet making a huge difference in overall progress.
Last but not the least, I am highly obliged and grateful to KTH LAB that has provided some rare/scarce laboratory resources enabling me to cover the scope and improve thesis material.

Abstract

In the era of distributed computing and multi-user environment, federated organizations though running in isolation with their own proprietary identity stores, need to collaborate and access each other's resources. Each of them has to authenticate its self to get authorization for the utilization of desired resources. Organizations use their own identity stores with user's credentials and policy enforcement mechanism for authorizing user access to their resources. In order to access resources of different organizations, a user must have login for all of them. This requires multiple identities for the same user which is very complex and difficult to manage. These conditions become even worse if collaborating organizations have used heterogeneous access control models for implementing their authorization policies. Existing centralized solutions such as Single Sign On (SSO) suffers with single point of failure and single central server could result in performance bottlenecks if not handled properly. Other distributed solutions for collaborating organizations require major infrastructure change and they also require homogenous access control model to be used between two collaborating organizations. In order to access resources user must be authenticated seamlessly and authorized to perform access request.

This research has proposed a plugin based distributed solution by making access control models, existing in different organizations, interoperable. The proposed solution has shown how decentralized and distributed yet federated organizations with heterogeneous access control models can share valuable resources/services in a secure, reliable and efficient manner with no or minimal changes to their existing infrastructure. The proposed solution converts the existing policies of collaborating organizations into Attribute Based Access Control Model (ABAC) by a Model Transformation Utility (MTU). When our proposed system is plugged-in to existing Role Based Access Control (RBAC) system, MTU reads RBAC policies form legacy repository and transforms them to ABAC policies using Extensible Access Control Markup Language (XACML) and stores them into ABAC policies repository. These policies are applied to remote request to obtain access over local resources.

In order to check the correctness of RBAC model transformation into ABAC model using XACML, a significant number of test cases have been designed, and executed on existing as well as transformed systems and the results comparison shows that model transformation is 100% correct.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter briefly describes evolution of access control model and challenges facing with advancement of distributed system. We highlight the motivation behind a distributed decentralized plugin based approach to achieve interoperability among access control models. In addition, our contribution in research and methodology used to validate implementation results is described.

1.1 Introduction

In order to protect the valuable resources, every organization needs access control mechanism that meets their requirements of managing their resources efficiently and effectively. Access control mediates user's attempts to access system resources by determining activities that are allowed to legitimate users. Access control can be implemented at various places and at different levels in an organization having information technology infrastructure. Operating system and database management systems (DBMS) need access control to shield their resources (files/directories in case of operating system and tables/views in case of DBMS) from inappropriate and unwanted user access. DBMS access controls are independent of those used by the underlying operating system.

Access control basic concepts includes subjects; an entity that may be person, device, process etc. objects; an entity containing information that is to be requested by subject and operations; an action subject wants to perform over object. Access control is concerned with three key principles of availability, integrity and confidentiality. System resources should be available when required and access control should have fault tolerance and recovery mechanisms to ensure incessant accessibility of system. Access control mechanisms should sufficiently be proficient to protect data from unauthorized access(Patil et al., 2007).

A typical access control deployment usually includes three logical components; an access control model, policies and a mechanism to enforce them. Policies in organizational context state which kind of users can perform what types of operations on a particular resource, in given circumstances. A formal representation of organizational policies is called access control model. Some very well-known access control model includes Mandatory Access Control (MAC); Discretionary Access Control (DAC); employed in most military organizations (Hu et al., 2006), Role-based Access Control (RBAC)(Sandhu et al., 1996); employed in commercial organizations such as banks that have large number of users and resources. These access control models are configured to organization's security framework(Vincent C. Hu, 2006). Bell La Padula Confidentiality Model (Bell and La Padula, 1976) and Biba Integrity Model (Biba, 1977) are early forms of MAC models. In MAC, system administrator assigns resource permissions to users and system objects are assigned security labels based on which access request is evaluated whereas in DAC each object has a list of permissions attached by the resource owner. DAC systems are fairly flexible as compared to MAC systems because permissions can be changed by adjusting resource security level from up to down and vice versa. Both MAC/DAC are the most common and oldest forms of access control models but their inherent limitation roused the evolution of new model based on user's roles called RBAC. In RBAC, roles are identified which may be an individual person's job or a group of users who may be identified by the organization hierarchy. Permissions are assigned to roles which are then assigned to users. RBAC is neither MAC nor DAC since the administrator controls the access at system level and permissions are also handled in a different manner. RBAC challenge is to balance the trade-off between strong security and easier administration (Hu et al., 2006). This deficiency was covered in Attribute Based Access Control (ABAC) model. Decisions made in ABAC are based on characteristics of subjects, objects, resources and the environment in which request is posed.

In this recent age of rapid growing internet due to advancement in network technologies, distributed systems gain incredible prominence in the field of information technology. A distributed system is a collection of autonomous system working together as single unit. These systems may be computers connected through network which enable them to share their resources and coordinate each other to accomplish related set of goals or it may be a software having processes running concurrently on different processors. Now a days, information sharing among federations and between enterprise sub systems has become common. These distributed systems require secure channels for sharing their sensitive information.

Federated organizations are de-centrally organized, semi-autonomous/ autonomous and may be geographically dispersed lines of business (LOBs), information technology systems and applications. Supply chain management and stock exchange systems in which multiple participants have established standard set of protocols for sharing information, are typical examples of federated organizations. Federated organizations merge or collaborate to provide services to their common users who want successful completion of their tasks in a secure way with minimal effort. Each collaborating organization needs to access its partner organization resources and vice versa. Therefore, users of remote organization should be authenticated and authorized to access the resources of partner organization. One possibility is to manage multiple identities for a user in both organizations, which is not only difficult to manage but is also troublesome for user to remember its multiple credentials. In order to make seamless authentication certain centralized authentication systems such as Single Sign On (Ltd, 2006) exist, which have their own limitations including single point of failure. Federated organizations also want their existing infrastructure to be unchanged or have minimal acceptable changes that does not affect their internal flows. In order to provide access to each other's resources, we propose distributed solution based on model transformation for collaborating organizations. Both organizations might have used heterogeneous model for implementing their access control requirements. Heterogeneity can be removed by using the same model at both sides which is not feasible for any organization to rewrite all its policies that might be in thousand in number. Our objective is to provide solution that requires no or minimal changes to existing infrastructure.

Our proposed plugin based solution include Model Transformation Utility which transform existing model into Attribute Based Access Control Model using XACML These policies are applied to remote request to give access over local resources whereas legacy policies are applied to local requests.

1.2 Motivation

Federated organizations collaborate or merge into one need of accessing each other's resources. User of collaborating organization should be authenticated seamlessly. User of collaborating organization should be authorized to manipulate requests from remote user (collaborating party). Existing solutions are centralized such as SSO, suffers with a single point of failure. Centralized server may be overloaded if more than one federation are included and other models of collaborating environments (described in the related work) require infrastructure changes. What we need, is a solution that requires no central server, to avoid a single point of failure and a distributed application with improved performance that requires minimal or no changes to the existing infrastructure (installed as plug-in). Authentication within an enterprise and across the organization must be seamless.

1.3 Problem Statement

The problem statement of the thesis is as follows:

"Provide a decentralized and distributed solution by making access control interoperable, for federated organizations (implemented different access control models) collaborating each other and accessing each other resources by authenticating/authorizing each other's users seamlessly."

1.4 Thesis Aims and Objectives

The main aims and objectives of this thesis are as follows:

I. Provide a decentralized solution for federated organizations based on notion of model transformation.

II. Realization of the proposed scheme by implementing it.

1.5 Thesis Contribution

In this research, we have solved the following issues and produced the subsequent contributions in the field of access control models.

1.5.1 Publication

This research work has been published with the title "***Interoperability among Access Control Models***" in IEEE proceeding of 15th International Multi Topic Conference (INMIC) at Riphah International University Islamabad, December 13-15, 2012.

1.5.2 ABAC Plugin

This work gives the notion of achieving interoperability among access control models by transforming them into an intermediate model; Attribute Based Access Control Model (ABAC). Both collaborating organization's existing access control models are transformed into ABAC by Model Transformation Utility (MTU), automatically. Transactions across the organizations are handled by new ABAC model where existing authentication and authorization flows are untouched. This plugin based decentralized, yet distributed solution requires minimal or no infrastructure changes and does not affect the internal workflow of adapting organization.

1.6 Methodology

Research methods are categorized into two broad approaches named as quantitative and qualitative. Research in natural sciences such physics, chemistry, geology, biology, etc. requires things to be observed and measured objectively and these investigations are worked upon by other researchers in the community. This process is termed as quantitative research. On the other hand, social sciences such as sociology, psychology etc. being concerned with the study of human behavior is difficult to explain in measurable terms. This phenomenon demands a new form of research called qualitative research. Qualitative research focuses on collection and analysis of subjective data which involves user's perceptions, whereas quantitative research is based on collection and analysis of objective data that can be measured in discrete values which may involve calculus and statistical data analysis. Qualitative research is inductive in nature in which we infer general from something specific whereas in qualitative research specific things are inferred from general. Quantitative research is about scrutinizing relations among predefined concepts while qualitative research figures out ideas from situations.

Although, software engineering is making rapid progress but still it is not mature enough to be called a pure engineering discipline. Traditional scientific and engineering models borrowed from other discipline such as sociology, philosophy and natural science do not best fit in an ever-growing field of software engineering. In order to solve software engineering complex and novice problems, researchers have introduced a few more models of arguments which include empiricism, hermeneutics, mathematical proof and proof by demonstration. In this research thesis, we adopt proof by a demonstration strategy in which an artifact (prototype implementation) stands as an example of proved hypothesis.

I. Background Learning and Literature Survey

A Number of existing Access Control Models including standalone and distributed access control models for both autonomous and federated organizations have been studied. Different types of markup languages, mostly based on XML, are studied in detail to choose the appropriate one for access request or response and implementation of policies. Legacy standalone access control model is mature enough to be implemented in some specific organizations whereas distributed access control models lack adaptability and backward compatibility which results in synchronization issues. Almost all of them require a major infrastructure change or replacement of legacy model at both federations in order to perform their joint operation. A widely used SSO approach requires central server for user authentication. Being a center dependent approach, it suffers with single point of failure which is its major drawback. Single central server could result in performance degradation because of congestion, if not handled appropriately. Therefore, a distributed solution for federated organizations was required that need to collaborate each other or may be merged into one.

Policy evaluation engines have also been keenly reviewed to choose the best implementation of XACML so that better performance could be achieved.

II. Problem Identification

By keeping in view the problems discussed in the above section, we found out that there should be some decentralized access control solution for collaborating federated establishments. A solution, which not only handles their enhanced collaborative needs but is also in compliance with their existing workflows.

III. Hypothesis

Can a decentralized yet distributed plugin based system for collaborating organizations (implemented heterogeneous access control models), accessing each other's resources through seamless authentication and authorization, be developed by making access control models interoperable?

IV. Determining appropriate tools and technologies

In this phase, tools and technologies available for implementing model transformation utility and authentication request and response have been described. XACL and XACML are languages that can be used for describing policies. SAML provides assertion that can be used for user's cross organizational authentication.

V. System Design and Architecture

Detailed system architecture which includes multiple modules such as Attribute Extraction Module (AEM), Rule Extraction Module (REM) and Model Transformation Utility (MTU) has been described. Moreover, request and response flow which include communication between local Identity Management System (IDMS) and remote IDMS and how the policies are generated and evaluated on user request, are described in detail.

VI. Implementation and Results

This section describes graphical user interface for AEM, REM and MTU modules. A test case example has been employed to show the complete authentication and authorization cycle. A number of test cases have been executed to compare system results before and after model transformation.

1.7 Thesis Organization

Rest of the thesis is organized as follow: Chapter 2 describes the overall background of the research area. Chapter 3 describes tools and technologies available for implementation. Chapter 4 discusses related work and overall review of existing solutions. Chapter 5 elaborates system architecture, design and workflow for the components of proposed system. Chapter 6 describes implementation and evaluation of system by giving example of complete request response cycle and Chapter 7 is the last chapter of the thesis. It concludes thesis work and highlights future directions.

1.8 Summary

As technology evolves inherent limitations of existing access control models rouse the need of new models. Most of the access control models such MAC/DAC, initially designed for military purpose, evolve into new models to handle challenges of commercial applications. Rapid growth of enterprise distributed yet collaborative application open new directions for system security. Enterprise single sign, being based of central server, inherits problems of performance bottleneck and single point of failure. In this research, we have concluded that plugin based distributed solution providing interoperability among access control models is better choice for collaborating organizations.

Chapter 2

Conceptual Knowledge

This chapter highlights the importance of access control model in our every-day life. Some of the well-known access control models, their strengths and limitations are described briefly.

2.1 Access Control

Access control, a very common phenomenon, enables an authority to control access over sensitive resources. A car lock, an ATM pin are means of access control. When a person or authority seeks to secure its confidential, sensitive and important information, access control possession is of prime importance. Some commonly used access control mechanisms are listed below:

- Discretionary access control
- Mandatory access control
- Role-based access control
- Attribute Based Access Control

2.1.1 Discretionary access control

In Discretionary access control (DAC) resource owner is responsible for resolutions of access policy. Owner decides the list of privileges is assigned to other users. DAC comprised of two major concepts:

- **File and data ownership:** Each resource in the given system has at least one owner. In DAC the object creator (subject) is an initial owner which determines its access policy.

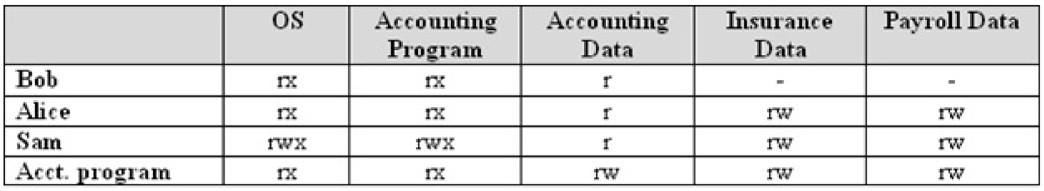
- **Access rights and permissions:** Owner can assign some or all control of specific resources to subject.

Access controls may be discretionary in ACL-based or capability-based access control systems. (In capability-based systems, there is usually no explicit concept of 'owner', but the creator of an object has a similar degree of control over its access policy.)

2.1.1.1 Access Control Matrix

Access control matrix proposed by Butler W. Lampson in 1971 is used for protection of system resources. ACM model restricts unauthorized access to resources by limiting user access right to specific objects. It is just like locked door which can only be unlocked only by the person having door key.

Access control matrix comprised of subjects, objects and operations that a subject can perform on object. Subjects (user or program) and object represents rows and columns of rectangular matrix which comprises data to symbolize access permissions. A sample access control matrix is shown in table below:



	OS	Accounting Program	Accounting Data	Insurance Data	Payroll Data
Bob	rx	rx	r	-	-
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Acct. program	rx	rx	rw	rw	rw

Table 2.1: Access Control Matrix

Bob, Sam and Alice (three users) are subjects and OS accounting data, Insurance data, and Payroll data are objects while accounting program act as subject and object at the same time.

Limitations

Access control matric performs well for small number of subjects and objects. As long as number of subjects and objects increases its complexity increases and performance is degraded. Large access control matrix also wastes too much memory.

In order to improve performance of authorization process and reduce system complexity access control matrix is spitted into rows and columns. Columns forms access control lists whereas rows take form of capability list(C-List).

2.1.1.2 Access Control List

Access Control list gained fame with emergence of multi-user system in 1970s. ACL's are most rudimentary form of access control mechanism. Access must be restricted to files and data on the shard systems in multi- user environment. In the beginning access control list were employed in UNIX systems but later on when multi-user operating system personal also adapted ACLs. In ACLs resources are referred as objects. Each object has an associating list which contain mapping between set of entities that can request access and set of actions that each entity can perform on it.

In order to understand it let us take an example of file system. Each file has an associated data structure containing list of user's along with multiple flag indicating access privileges such as read, write, execute, delete, modify, append etc. or any combination of these. This ACLs file is consulted by operating system whenever a user tries to perform some sort of operations to the given file. This concept may also be applied to group of objects such as directory or application/system processes and group of processes.

ACLs are relatively simple as it does not require technological infrastructure for implementation. It is not only adapted in early basic systems but also implemented in today's modern operating systems such as UNIX. Developers can also use ACLs at application level to control user access at interface level. This can be achieved by using map- type data structure such as data dictionaries in Python language and their counterparts Maps in JAVA programming languages. Databases may also be used to store ACLs in certain condition such as number of users in an organization is too large for in-memory data structure to scale well.

ACL Composition

Tag Type: ACL entry is stated using tag type. For example owner of the file, group owing file, user name, group name or others. (for all other users)
Qualifier field: Tag type instances are specified in qualifier field. Userid and groupid are specific names for named user and named groups respectively.
Permission Set: Access rights such as read, write, execute etc. are specified in permission set. ACL extraction for insurance data from access control matrix mentioned in table 2.2 is shown below:

Example	**Description**
user:rwx	File owner has read, write and execute privileges as ACL contains file owner tag with qualifier field empty.
user.userid:rw-	Userid has read and write privileges as qualifier userid is specified by named user tag.
group:r-x	Owning group has read and execute privileges as ACL contains group owner tag with empty qualifier.
group.groupid:–x	Groupid has read privileges as groupd name is specified with group name qualifier.
other:r–	All users other than owing user, named user, group user have read privileges.

Table 2.2: ACL Examples

(Bob,-),(Alice,rw),(Sam,rw),(Acct. Program, rw)

Limitations

ACLs are not an efficient mean for access control as it must be checked every time whenever a particular file, process or other resource is accessed.

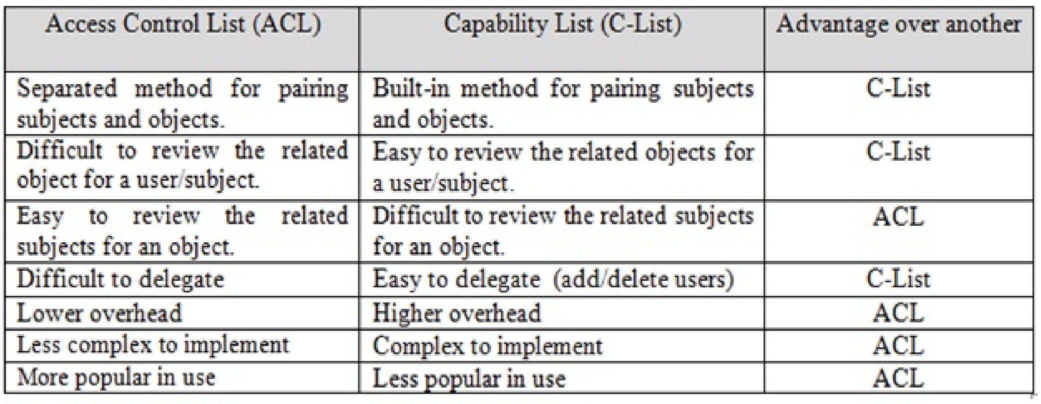
ACLs are not only used to control user access over system resources but it is also used to control access of application processes access over system resources. Each time a user need access it perform ACLs lookup, applications invoked by user needs resources, also perform lookup and application invoking application and processes also need to perform lookups and so on.

In an enterprise where a large number of users or group of users requires various levels of access permission, ACL can be very difficult to manage. Adding, removing, updating user's privileges not only a time consuming process but it can also result in inconsistent or error prone system if not managed properly.

As ACLs are resource centric and are usually written by owner or initial creator of resources. So a strong level of coordination is required between users and resource owner which is quite cumbersome when users and resources are large in number. [2]

In an ACL system, we can easily find the list of users having access to that particular resource but it is nearly impossible to find list of privileges a particular user for system resources.

2.1.1.3 Difference between Access Control List and Capability List



Access Control List (ACL)	Capability List (C-List)	Advantage over another
Separated method for pairing subjects and objects.	Built-in method for pairing subjects and objects.	C-List
Difficult to review the related object for a user/subject.	Easy to review the related objects for a user/subject.	C-List
Easy to review the related subjects for an object.	Difficult to review the related subjects for an object.	ACL
Difficult to delegate	Easy to delegate (add/delete users)	C-List
Lower overhead	Higher overhead	ACL
Less complex to implement	Complex to implement	ACL
More popular in use	Less popular in use	ACL

Table 2.3: ACL VS. C-LIST

2.1.2 Mandatory access control

In Mandatory access control (MAC) system administration is responsible for writing access control policies for resources instead of resource owner. Government and military multilevel systems use MAC in order to secure their classified or highly sensitive data. A multilevel system may be single computer system handling several classification levels concerning subjects and objects.

Sensitivity labels: All subjects and objects are assigned labels in MAC based systems. Sensitivity labels are used to specify level of trust. A subject's sensitivity level shows its level of trust whereas object's sensitivity level shows the level of trust required by requesting subject in order gain access. Subject having sensitivity level must be equal or higher to object sensitivity level to accomplish its access request.

Data Import and Export: In MAC based systems sensitivity labels should be properly maintained and implemented while importing information from other systems or exporting information to other systems. Import and export should be controlled enough to protect sensitive information in the entire process.

Two methods are commonly used for applying mandatory access control: Rule based Access Control and Lattice based Access Control.

2.1.2.1 Rule based (or label based) access control

Rules are implied for granting access to the object requested. All rule-based MAC system evaluates simple rules to decide whether access is granted or denied. Rule based system can be combined with other access control models such as Role Based Access Control (RBAC), DAC etc.

2.1.2.2 Latice-based access control

Lattice based access is used for complex decision making which involve numerous subjects and objects. Lattice model comprise of mathematical structure which include pair of subject and object. Greatest lower bound and least upper of these pair is also defined in lattice based access control model.

2.1.3 Role-Based Access Control Model

RBAC mechanism is used to define complex access control policies in most commercial applications such as banks that have large number of users and resources. RBAC model is based on roles to which privileges and users are assigned. A role may be person's job function or a group of user having same set of privileges. However, in some system there is a slight difference between concepts of groups and roles. Groups are typically described as the set of users and they can be assigned a single role multiple roles.

RBAC access control policy is composed of relationships between roles and permissions, users and roles, and role and role. Administration complexity can be reduced by assigning roles to users, permissions to roles and defining role hierarchy. RBAC also some well-known security principles such as least privileges (only those permission are assigned to roles which are required by member to perform required task), separation of duties (mutually exclusive role can be invoked to complete a sensitive work flow). The NIST defines RBAC in two ways:

- Core RBAC
- Hierarchical RBAC

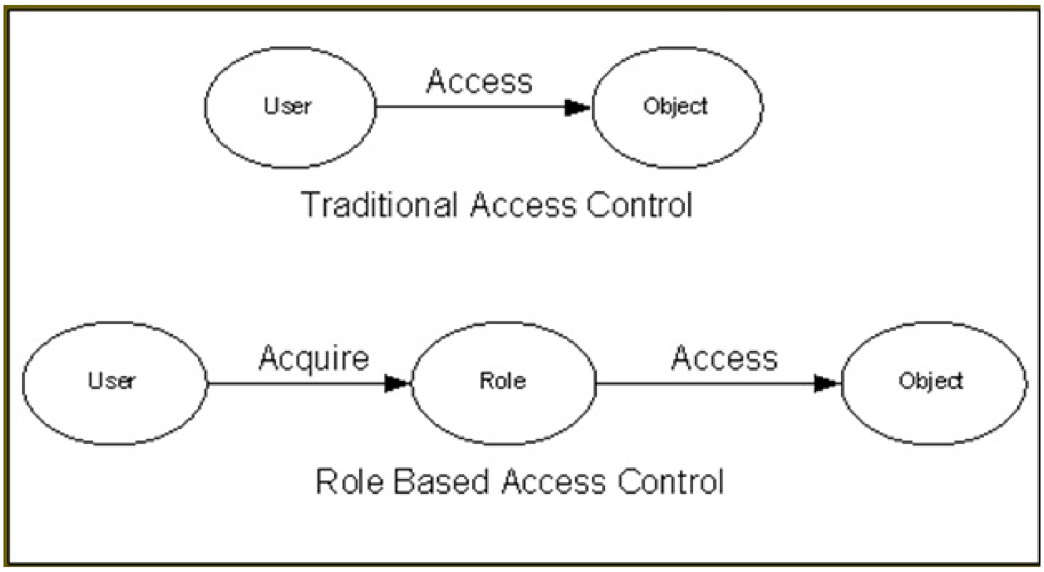


Figure 2.1: RBAC vs Tradional Access Control Model

2.1.3.1 Core RBAC

Core RBAC model shields only the vital features of basic RBAC model. In core RBAC model there can be many to many relationship between user and role and permission and role whereas in original RBAC users are assigned to roles and permissions are attained being a member of roles. Figure 2.2 describes core RBAC model in which user are assigned roles and permissions (operations that can be performed on objects) are assigned to roles. Roles can be activated and deactivated by enclosure of concept of users and roles sessions. Core RBAC allows one user to have multiple roles as well.

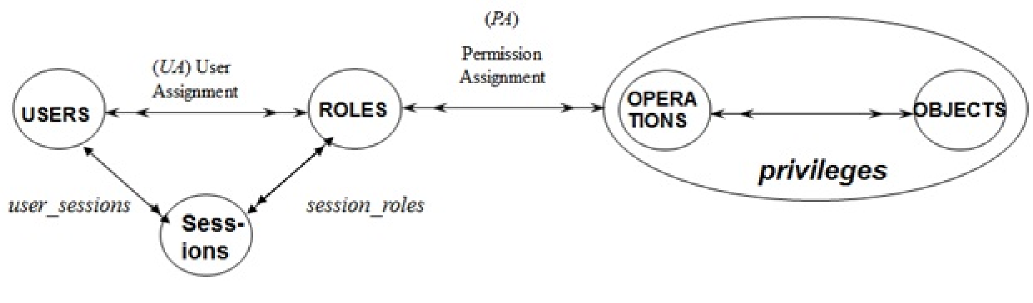


Figure 2.2: Core RBAC

User: A process or a person.
Role: Authority, Job function.
Operations: Program specific function's execution.
Objects: An entity also called resource which needs protection.

2.1.3.2 Hierarchical RBAC

Roles can be overlapped with other roles and a role can be subset of other role. For example a specialist is also a doctor and intern. Roles hierarchies are introduced to define seniority relationship among them. Senior role automatically acquires permissions of junior roles. Role/role relationship defines

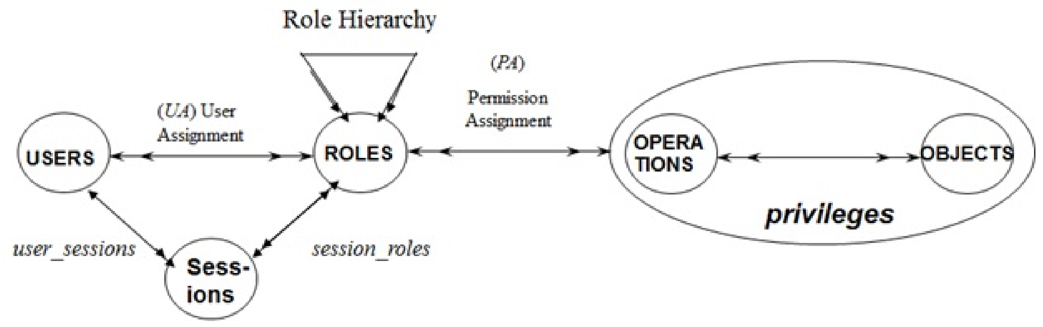


Figure 2.3: Hierarchical RBAC

privilege inheritance and user's membership. Role hierarchies also reflect organizational structure and functional descriptions. Both Dermatologist and cardiologist are specialist doctors and doctor is also an employee of hospital. Membership and privileges are shown in figure 2.4.

Limitation

RBAC challenge is to balance the trade-off between strong security and easier administration [1]. In order to make system more secure roles need to be defined at more granular level thus in result a single user can have multiple roles which are very complex to manage. Numbers of roles should be minimized in order to make easier management.

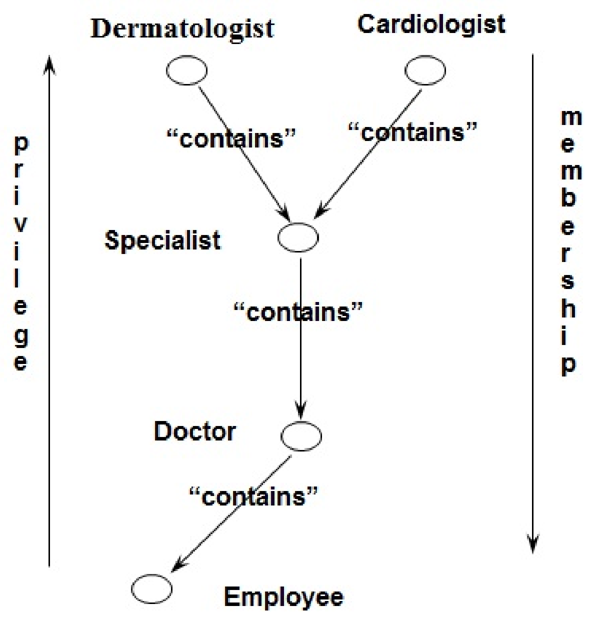


Figure 2.4: Hierarchical RBAC(Membership vs Priveleges)

2.1.4 Attribute-based access control

Attribute Based Access Control (ABAC) uses attribute information associated with subject and resources in question. A typical implementation of ABAC includes subject which demands access, resource on which access is required, action and the environment (the context in which request is made).

In attribute-based access control (ABAC), access is granted on the basis of user's attributes rather than the rights of subject associated after authentication. The user has to prove claims about his attributes to the access control engine. An attribute-based access control policy specifies which claims need to be satisfied in order to grant access to an object. For instance the claim could be "older than 18" and any user that can prove this claim is granted access. Users can be anonymous as authentication and identification are not strictly required.

In ABAC model, attributes of user requesting object, request environment and resource in question's are considered for making an access control decision. Attributes are compared with set of predefined values in order to allow or deny a request. Attribute may not be related to each other and come from various sources. For example consider the scenario of organization employer;

hiring date of project on which he is working, his role in an organization and his station of posting are some of the attributes.ABAC component include four entities:

Subject: A user or an application etc. originating request.
Resource: An object for which request is made.
Actions: An operation that a user wants to perform on a particular object.
Environment: Specify the context in which request is made.

2.2 Summary

In our daily life we use access control mechanism to protect our belongings from an authorized manipulation. In the same way information resources also require mechanism for protection from an unauthorized access. Some very well-known access control model includes Mandatory Access Control / Discretionary Access Control (employed in most military organizations) and Role-based Access control (employed in commercial organizations such as banks that have large number of users and resources). These access control models are configured to organization's security framework and they sometimes need to be tailored for particular organizations domain perspective. ABAC model being context aware model considered as new generation access control model that can replace all of the traditional access control models.

Chapter 3

Tools & Technologies

This chapter covers different languages that can be used to write access control policies and also provide a mechanism for complete authentication and authorization request and response cycle.

In this global world of information technology, enterprise applications are gaining immense prominence as time progress. An enterprise usually has large number of resources that must be protected using some sophisticated access control policies. Access control security policies may have multiple enforcement point and a huge number of elements that require a separate department. In order to implement accurate security policies each enforcement point is configured independently which is very expensive and highly unreliable when policy modification or addition is required. Meanwhile consumers and enterprise shareholder want corporate and government executives to establish best practices for securing enterprise intellectual assets. Therefore, a common policy language is required to be implemented in the enterprise so that all of the enforcement elements of enterprise security policies are managed through enterprise information system.

In order to express security policies of information system a policy language must have the following capabilities:

- Policy language should have a mechanism to combine rules in policies and policies into policy set which can be applied to a specific decision request.
- Policy language should have capabilities of dealing with multiple subjects.
- Policy language should provide a method to devise an authorization

decisions based on subject and resource attribute.

- It should have support for multi-value attributes.
- A comprehensive set of mathematical and logical operators must be supported by policy language that can be applied to manipulate subjects, objects and resources.
- Policy language should capable to handle distributed policy components.

XML, having easy to use syntax and semantic capabilities make it suitable for writing security policies. XML provides extensive support for multiple platforms and it can also be enhanced to meet specific requirements of particular application in its domain context. Some of the XML based solutions meeting above criteria has been listed below:

3.1 XML Access Control Languages (XACL)

XACL is first XML based access control language that is designed and developed by IBM. It not only provides a sophisticated access control model but also provides a policy specification language as well. This model is used to implement access control policies.

Objects: is a resource in question.
Subjects: may be user or group or may be a process etc.
Actions: operations such as read, write, create and delete that a subject want to perform on object.
Conditions: that must be true for successful completion of request.

3.1.1 Limitation

XACL architecture is limited to provide access control to XML document and has very low express ability. It is not best suited for distributed environment as it is manage centrally.

3.2 Extensible Access Control Markup Language (XACML)

Extensible Access Control Markup Language (XACML)(Simon Godik, 2003; Tim Moses, 2005; Erik Rissanen, 2010), is an OASIS (OASIS, 2013) standard of policy specification language. The policy and the decisions that are made on requests by applying policies and the end response are encoded in XML. General access control requirements are met using policy language and it can be further enhanced to define new data types, function and policy/rule combining algorithms. XACML engine can be queried by sending request and corresponding response can be one of the four values: Permit, Deny, Intermediate (decision can't be made due to some error and misinformation) and Not Applicable (said service can not answer the request)(Microsystems, 2004). XACML data flow model components details are listed below:

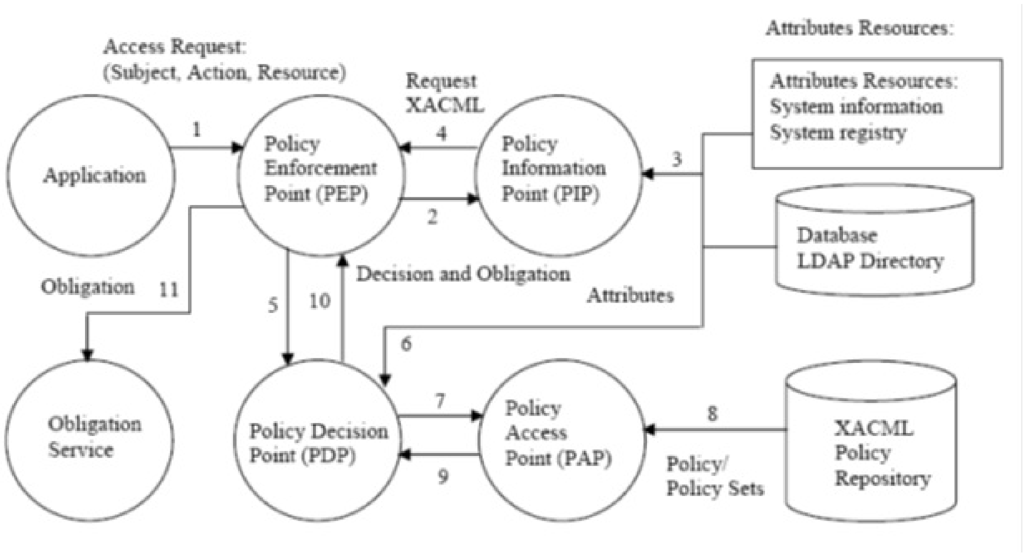


Figure 3.1: XACML Architecture

Policy Administration Point (PAP): Policy Administration Point (PAP) is responsible for writing XACML policies and policy sets. Policy sets and policies matching a particular target are made available to PDP (Policy Decision Point) for evaluation.

Policy Enforcement Point (PEP): User, application or any process sends access request to PEP. PEP using context handler prepares request in its native format which may include subject, resource, action and environment attributes. It can request Policy Information Point (PIP) in case additional attribute are required.

Policy Information Point (PIP): Policy information point provides additional request attribute to PEP.

Policy Decision Point (PDP): PDP is responsible of evaluating request by applying policies and returns back appropriate decisions to PEP.

XACML typical implementation includes Policy enforcement Point (PEP), to which request is made. It formulates the request in the context of resource, action and environment attributes and related to requester and resource in question, and forwards this request to Policy Decision Point (PDP). PDP finds applicable policies and returns the appropriate response back to PEP according to applicable policies. XACML context is shown in figure 3.2.

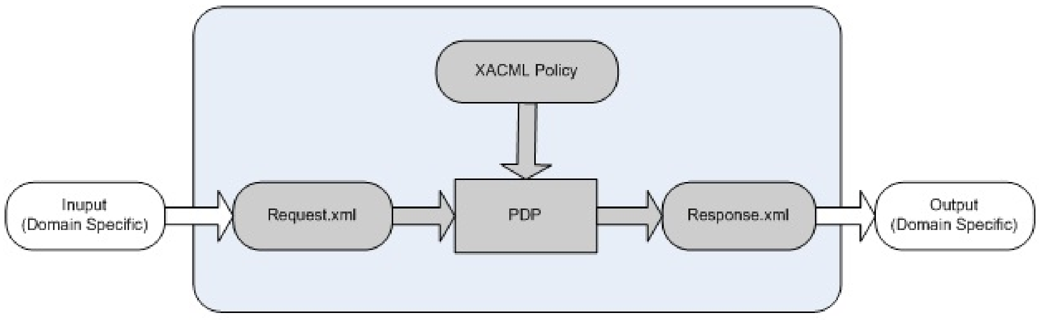


Figure 3.2: XACML Context

The main components of XACML engine include Policy, Policy Set, Targets, Rule, Rule Target, Rule Combining Algorithms and Policy Combining Algorithm. XACML policy structure is shown in figure 3.3.

PolicySet: Policy set includes the multiple related policies.

Policy: Policy contains a set of rules including default rule which is applied when no rule is applicable.

Target: Target is used to define set of subjects, actions, resources and environments to which policy or rule is applied. It is optional element in rules.

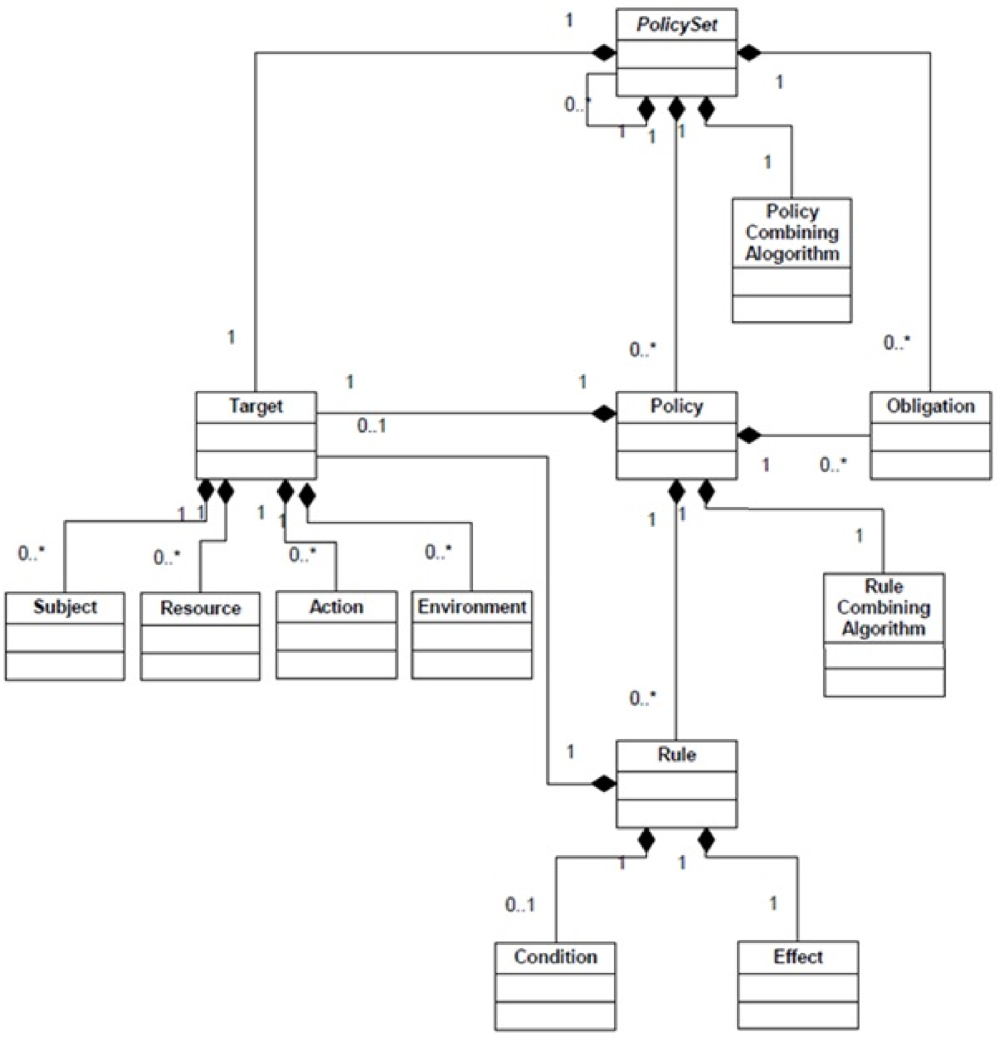


Figure 3.3: XACML Policy Structure

Rules: Rules are encapsulated by policy. It cannot exist in isolation. The main components of rule include target, conditions and effect. If condition is true it returns effect value which can be either true or false. If condition is not applicable it returns not applicable and in case of error or missing data intermediate result is returned.

Rule Target: Rule target which can include subject, resource, action and environment attribute to decide on whether rule will be applied or not. Request that matches the rule targets are tested against the rule and return the appropriate response (permit or deny).

Rule Combining Algorithm: Result of multiple rules are combined using rule combining algorithm. If a policy contains combining parameter it effects the operation of rule combining algorithms.

Policy combining algorithm: Policy combining algorithms are used to combine individual results of multiple policies.Standard combining algorithms include:

- Deny-overrides (Ordered and Unordered)
- Permit-overrides (Ordered and Unordered)
- First-applicable
- Only-one-applicable

First version of XACML was introduced in August2003; later on enhanced version of XACML V.2.0 was released in February, 2005. Up till now XACML V.3.0 has been released on August, 2010.

3.2.1 Differences between XACML versions 1.0 and 2.0

(Daniel, 2005)

3.2.1.1 Context Schema Changes

(OASIS, 2005a)

- XACML 1.0 namespace urn:oasis:names:tc:xacml:1.0:context is replaced with urn:oasis:names:tc:xacml:2.0:context:schema:cd:04.

- XACML 1.0 request contain only one resource whereas XACML 2.0 allows more than one resource in request.
- Environment element which was optional in XACML 1.0 made mandatory in XACML 2.2.
- XACML 2.0 allows more than one attributes values whereas XACML 1.0 supports only one attribute value.
- In XACML 2.0 it is optional to have status element in result while it was mandatory in XACML 1.0.
- The IssueInstant attribute that appeared optionally on Attribute elements in XACML 1.0 has been phased out in XACML 2.0.

3.2.1.2 Policy Schema changes

- In XACML 2.0 namespace urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04 has been replaced with XACML 1.0 old namespace urn:oasis:names:tc:xacml:1.0:policy.
- CombinerParameters which can be of any number are introduced in XACML 2.0.
- XACML 2.0 introduced optional element version having default value equal to 1.0 in policy and policy set.
- In XACML 2.0,PolicySetIdReference and PolicyIdReference type has been changed to from xs:anyURI to xacml:IdReferenceType.
- A new element named VariableReference is introduced in XACML 2.0.
- RuleId attribute of rule elementhas been changed to xs:string in XACML2.0 while it was xs:anyURI in XACML 1.0.
- An EnvironmentMatch element has also been introduced in XACML 2.0.
- Subjects, Resources and Actions elements has been made optional in XACML 2.0.
- AnySubject, AnyResource and AnyAction elements have been removed in XACML 2.0.
- AttributeValue ,VariableReference ,AttributeSelector, FunctionApplyand all FooAttributeDesignator elements have been introduced in XACML 2.0 being members of Expression group.

3.2.2 Differences between XACML versions 2.0 and 3.0

(OASIS, 2005b)

- Obligation can be included in rule in XACML 3.0 whereas it was only added in policy sets and policies in XACML 2.0.
- XPath expression can be applied to content element root in XACML 3.0 while in XACML 2.0 it can be applied to XACML request root.
- Disjunctive and Conjunctive function of the target group category (e.g. Subject) have been removed in XACML 3.0 and new elements AnyOf and AllOfhave been introduced in XACML 3.0.
- User can create customized categories in XACML 3.0 while in XACML 2.0 defines fix categories (Subject, Resource, Action and Environment).
- A new element called Advice has been introduced in XACML 3.0 that is similar to obligation.(OASIS, 2010)
- Two new profiles “XACML 3.0 Export Compliance-US (EC-US) Profile Version 1.0” and “XACML v3.0 Administration and Delegation Profile Version 1.0” has been introduced and ”XACML v2.0 Multiple Decision Profile Version 1.0” has been updated in XACML 3.0.

3.3 Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML)(OASIS, 2005c) is also an XML based framework of OASIS standard that is used to communicate user’s attribute information for authentication purpose. SAML wrapped existing technologies instead of introducing new technologies. SAML offers varieties of profiles to cover various interoperability use cases. These profiles can further extended to cop organizations specific security requirements. SAML assertions are made for attributes, identity and user entitlement in federated organization. In order to understand how SAML works, we first introduce some terms commonly used in SAML.

Asserting Party: is an entity which generates SAML assertion.

Identity provider: an entity which not only creates maintains and manages principal’s information but also provide authentication to service provider.

Relying Party: is an entity that takes decision based on information provided by identity provider.

Service provider: is an entity that provides services to principal or other entities.

SAML Assertions: SAML assertions are series of statements structured about a subject. They are means to make a claim about someone.

SAML assertion can be an:

Authentication statement: For example “Khalid authenticated with a smartcard PKI certificate at 8:05pm today”

Attribute statement: can contain multiple attributes such as “Khalid is a manager and has a 70000 spending limit”

Authorization decision statement: used to made decisions for example “Yes, Khalid can download that web page”.

SAML is flexible enough to extend and customize as needed. We use it in our proposed system, to authenticate partner organization’s user accessing service at service provider’s end. SAML Assertion statement structure is shown in figure 3.4.

3.4 Summary

A versatile policy specification language should have mechanism to combine policies and rules into policy and policy set respectively and a rich set of mathematical and logical operators. It should be able to handle the distributed policy components with support of multi-value attributes. XACML is very rich and comprehensive language as compare XACL. SAML can be used for authorization purpose but it is considered as best in asserting statement for authentication and querying attributes.

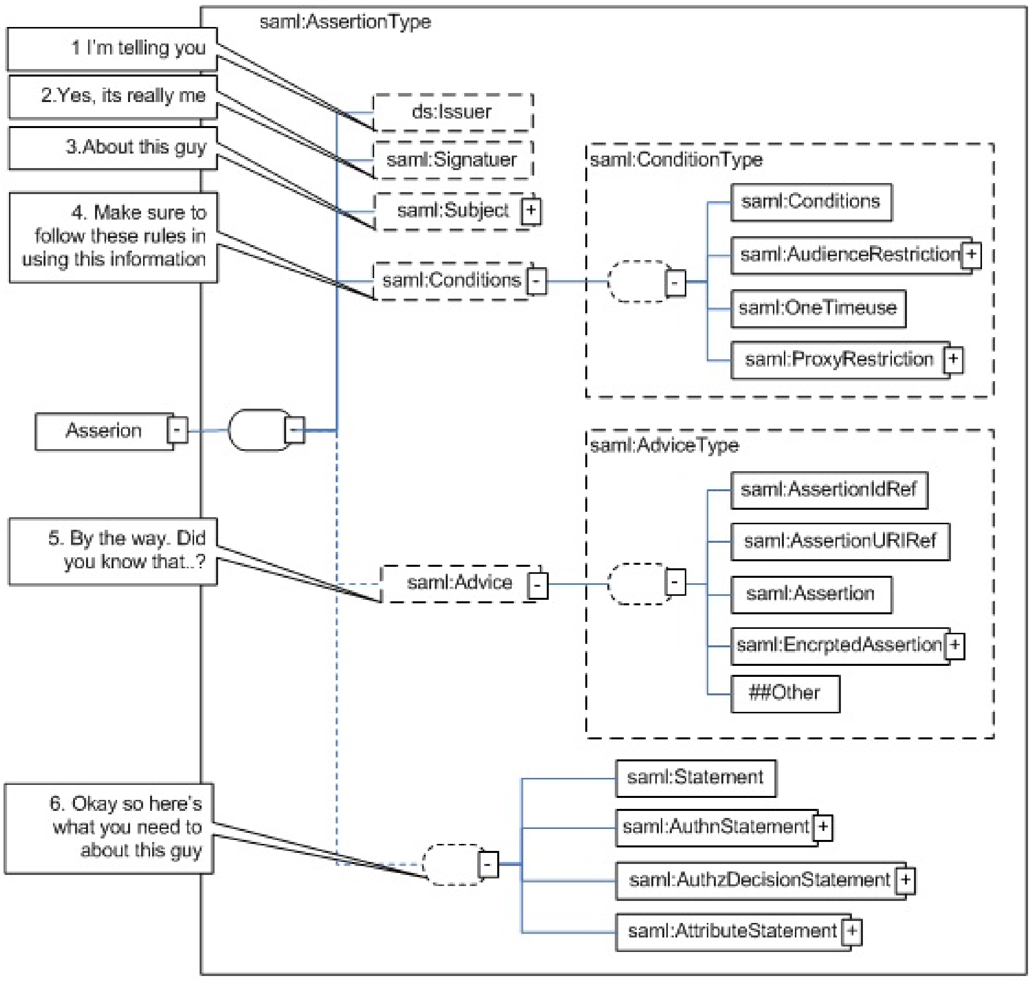


Figure 3.4: SAML Assertion Structure

Chapter 4

Related Research

This chapter covers different existing access control models employed in distributed environment. Some of these standalone access control models are enhanced to cop distributed environment needs. In addition, multiple implementations of XACML engine are compared to choose best policy description language. In the last section of this chapter existing solutions and their drawbacks are also listed.

4.1 Distributed Access Control Models

4.1.1 RBAC family frameworks

The access control decisions in the RBAC family frameworks (i.e., TMAC (Thomas, 1997), TBAC(Thomas and Sandhu, 1998), CBAC(Cohen et al., 2002)) are based on set membership queries. That is, when these models try to address collaborations in a distributed environment they need to rely on a mechanism that communicates internal state of collaborating domains to all collaborators or a mediator if the collaborations are mediator facilitated. These models facilitate collaboration across domains and have RBAC as their underlying access control model. Team Based Access Control (TMAC) refers team as a group of collaborating users acting in various roles and provides a way to assign permissions to the team whereas Task Based Access Control (TBAC) synchronize access permissions with ongoing tasks and workflow instances spanning across organizations.

4.1.2 Attribute Based Access Control and Security for Collaboration Environments

Jian et al (Zhu and Smari, 2008) compare different access control models and the Active Security Management and Distributed Security Management in terms of scalability, flexibility, granularity, ease of use and ease of management. In this paper distributed environment has been considered as secure group communication and two basic architectures have been discussed: a layered architecture and integrated architecture. Layered architectures are more robust while integrated architectures are more complex however. A distributed attribute based access control model has been introduced that assign permission on the basis of privacy preservation scheme and level of trust.

4.2 XACML for Access Control in Distributed Systems

Markus et al (Lorch et al., 2003) presented XACML as a distributed framework and also discussed how authorization can be deployed in decentralized systems, This paper discusses how XACML can effectively solve problem for existing authentication and authorization system such as Shibboleth (Scott Cantor, 2003), Cardea (Lepro, 2004) and privilege and policy management in PRIMA(Lorch et al., 2004) system. Secondly, authors have suggested certain points such as creation and management of access control policies, encoding of privilege management policies in XACML, locating the correct PDP, XACML request preparation and request context management that must be considered while using XACML for addressing challenges of distributed authorization. In the end authors conclude that XACML is an excellent choice for distributed authorization because of its ability to handle decentralized polices.

4.3 XACML Policy Evaluation Engine

Alex et al (Liu et al., 2008) has proposed a scheme called XEngine which efficiently evaluates XACML policies as compared to Sun PDP. In order to improve the processing efficiency tree structure policies have been used and found out XEngine performance is quite good whether number of rules in polices is small or large.

Nuo Li, JeeHyun Hwang, Tao Xie (Li et al., 2008) have compared the different implementations of XACML by testing input (XACML policies and requests) and observing output (XACML response). About three hundred and seventy four pair of XACML policies and requests has been used to test these implementations. Each pair of policy and request contains a particular XACML standard functionality. Among these implementation XACML.NET fails in supporting 34 of those functionalities and SUN XACML 1.2 fails to support 11 functionalities.

4.4 Single Sign On

Single Sign On (SSO)(Ltd, 2006) enable users to logon into enterprise's various applications using the same credential. Federated entities when merges or collaborates to perform some sort of joint operations, also adapt single sign on phenomenon for their authentication process. For example a business partner employee clicks on link of another federated enterprise; he must be authenticated to gain access over enterprise's resources. In this case access request is routed towards single sign on service for authentication that provides a security assertion token (using protocol like Shibboleth and SAML(Provost, 2009)) to enter into enterprise system. SSO established end to end audit session for user in order to improve security reporting and audits. It also enables uniform authentication and authorization policies through the entire enterprise application.

4.4.1 Limitations

SSO implementation include centralized server through which authentication request is routed that result in single point of failure. If Single sign on system fails, no user can access resources under the protection of SSO system even if applications are working. Centralized server may be overloaded when one more than one federations are included.

4.5 Shibbloeth

Shibbloeth(Scott Cantor, 2003) an open source software system implements web single sign on use case for user's authentication within an enterprise and across the organization boundaries. Shibboleth promises security and privacy preserving while exchanging user's information among organizations. It allows users to execute operations at an institution other than that which authenticate it.

Shibboleth includes four key concepts:

Identity Providers(IdP): Component running at organization whose user wants to perform operation on protected resource.

Service Providers(SP): Component running at protected resource provider end.

WAYF Service: Where Are You From service is located between federation entities or enterprise modules identity provider and service provider.

Federations: Shibboleth federation is trusted group of institutions agreeing on common authentication practices and policies. User sends access request

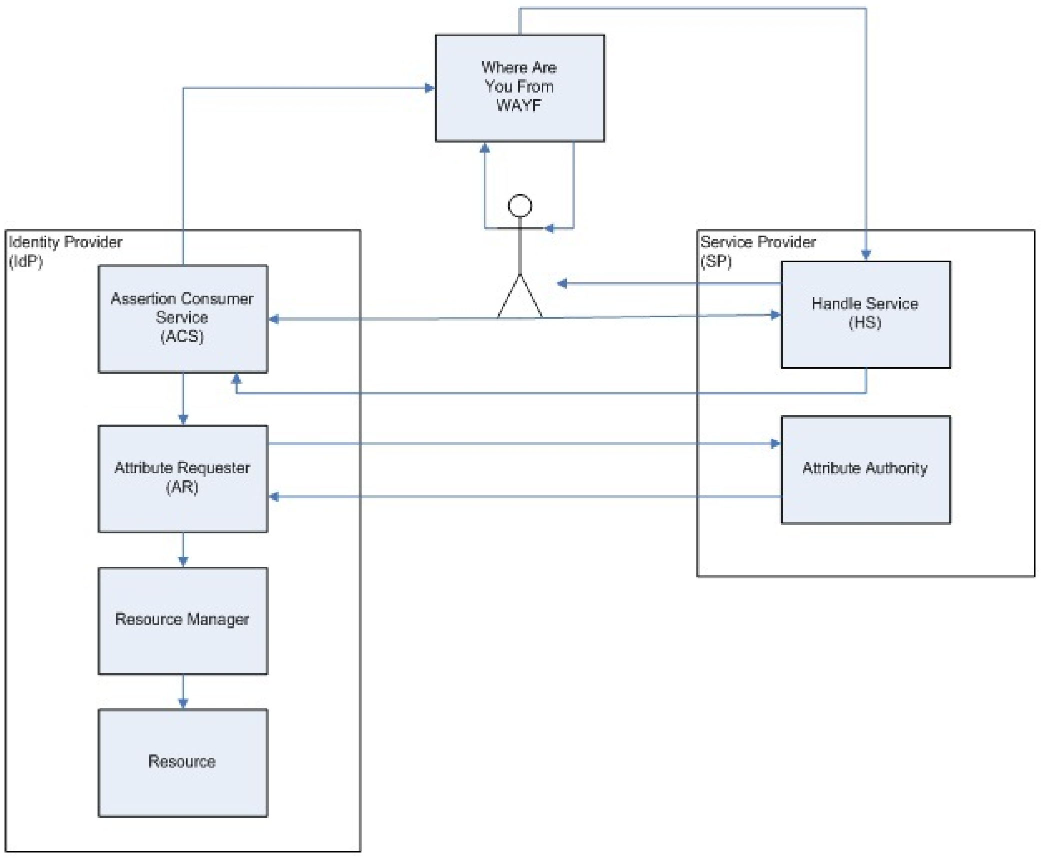


Figure 4.1: Shibboleth Component Interactions

to service provider's assertion consumer service which redirect request to Where Are You From (WAYF) service for authentication. WAYF service ask user about its organization IdP for authentication and redirect user to IdP's handle service. Handle service in conjunction with local single sign on validate user credentials and supplies handle to Assertion Consumer Service (ACS) of Service Provider (SP). ACS create session after validating handle and transfer it to Attribute Requester (AR) which request attributes from IdP's Attribute Authority(AR). Attributes provided by AR are used to determine access request to resource at SP end. User can only be able to access resource if permission is granted.

4.5.1 Limitation

Shibboleth components include centralized WAYF service through which authentication request is routed so it can suffer with single point of failure. If WAYF service fails, no user can access resources under protection of Shibboleth system even if applications are working.

4.6 Summary

Distributed access control models of RBAC family frameworks (TBAC,TMAC and CBAC), are enhancement of core RBAC model. Each of the above models requires major infrastructure changes for adaptive organization.
XACML, a policy and request/response language is an excellent choice for distributed authorization because of its ability to handle decentralized polices.
Sun's XACML implementation is relatively better than its other counterpart XACML.NET.
Shibboleth centralized WAYF service for single sign on inherits single point of failure problem for being centralized in nature.

Chapter 5

System Design & Architecture

This chapter describes systems high level as well as detailed technical deign of proposed system. System's various components/module and their function are described in detail. In addition to this workflow of the system are also discussed with the help of example.

This proposed system works as a plug-in to existing system. Figure 5.1 shows the high level view of the system with multiple modules such as Identity Management System (IDMS), Model Transformation Utility (MTU) and transformed policies in ABAC model. These modules work together to provide a workable solution that makes heterogeneous access control models of collaborating organizations, interoperable. Each of the collaborating organizations needs to access its partner organization resources (services etc.) and vice versa. Users of remote organization should be authenticated in partner organization to gain access over its resources (services).

In order to understand the entire working of each module let us consider two different entities say a software company and a university department works in collaboration for certain research projects.

Software company experts may offer courses for university students and may also send their trainees to enroll for university offered courses in order to enhance their knowledge in new research areas. Both institutions have implemented an access control model that best suits their interest. Suppose university follows RBAC model and Software Company uses ACL. Both of them shares their organizational hierarchy and agrees upon membership level mapping between roles and ACL groups as shown in table 5.1

When a Software Company (Identity provider) user wants to access a re-

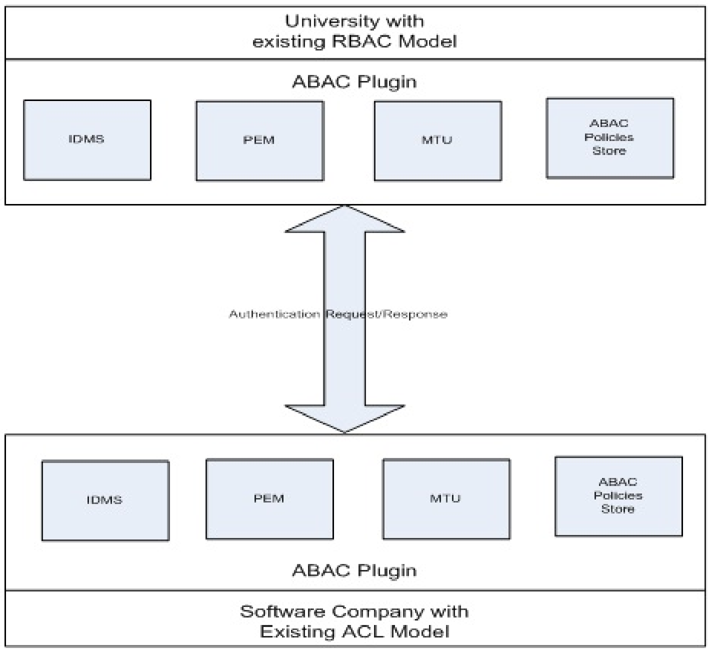


Figure 5.1: High Level Architecture

Software Company	**University Department**
Trainee	Student
Associate SE	Teaching Assistant (TA)
Software Engineer (SE)	Lecturer
Senior SE	Assistant Professor
Advisory SE	Associate Professor
Senior Advisory SE	Professor

Table 5.1: Mapping

source at university (Service provider), it sends request to university IDMS. Figure 5.2 shows a Software Company employee (Trainee) accessing University resources (say Course 'X').

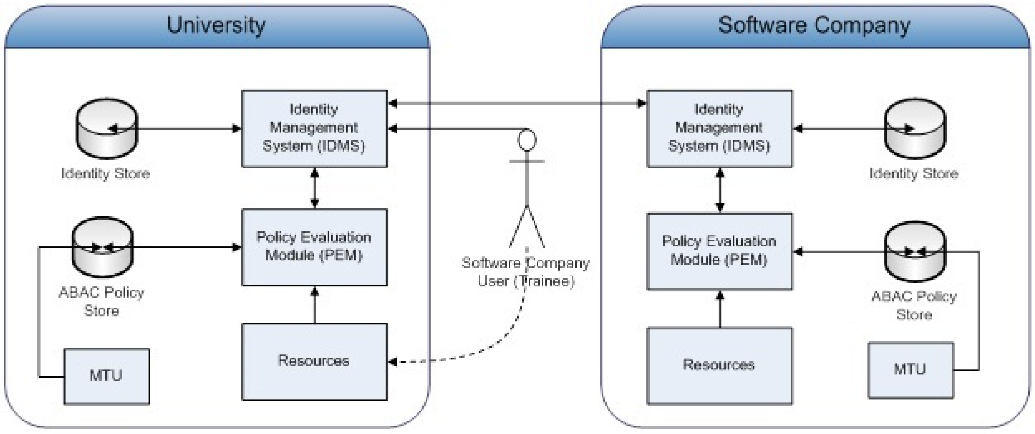

University
Identity Store
Identity Management System (IDMS)
ABAC Policy Store
Policy Evaluation Module (PEM)
MTU
Resources
Software Company User (Trainee)
Software Company
Identity Management System (IDMS)
Identity Store
Policy Evaluation Module (PEM)
ABAC Policy Store
Resources
MTU

Figure 5.2: User Scenario

5.1 Identity Management System (IDMS)

Identity management system is responsible for user account creation, management and teardown. User account creation comprises assigning appropriate level of access to resources in order to perform their respective operations. User account management comprises of updating of user's information and tuning of their access level as when required. Account teardown involves deactivating of user accounts which are no longer affiliated with organization.

When a user wants to access system resources, it sends authentication request to IDMS in order to prove its identity. IDMS first analyze request either it is coming from local (University) user or from remote (Software Company) user. If it is from local user, IDMS authenticates user and act as an Identity Provider (IdP) for providing user level access to authorization system otherwise IDMS uses SAML assertion for authenticating a remote user from its identity provider. When a request from remote user (Software Company) is received, university IDMS forward this request (SAML authentication request) to IDMS of Software Company which evaluates it and sends response in the form of SAML assertion. Figure 5.3 shows how SAML authentication request is validated and the response is conveyed back that includes other required attributes such as membership level. IDMS on

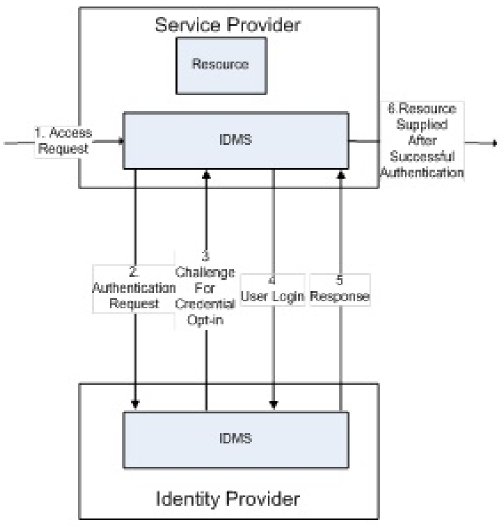

Service Provider
Resource
IDMS
1. Access Request
6.Resource Supplied After Successful Authentication
2. Authentication Request
3 Challenge For Credential Opt-in
4 User Login
5 Response
IDMS
Identity Provider

Figure 5.3: IDMS (Local) to IDMS (Remote) Communication

successful authentication stores user attributes in repository and forward access request to Policy Enforcement Point (PEP) of Policy Evaluation Module for authorization.

5.2 Policy Evaluation Module (PEM)

Policy Evaluation Module consists of two components: Policy Enforcement Point (PEP) and Policy Decision Point (PDP). PEP is responsible of performing access control by making decision requests and enforcing authorization decisions. PEP prepares request in native format (XACML request) using context handler and forwards to PDP for evaluation. In case of remote request, PDP consults mapping entries for membership level and applies policies obtained from ABAC policies store as shown in figure 5.4. ABAC policy store contains the policies which were previously converted by Model Transformation Utility from legacy model (RBAC) to ABAC.

5.3 Policy Administration Point (PAP)

PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.

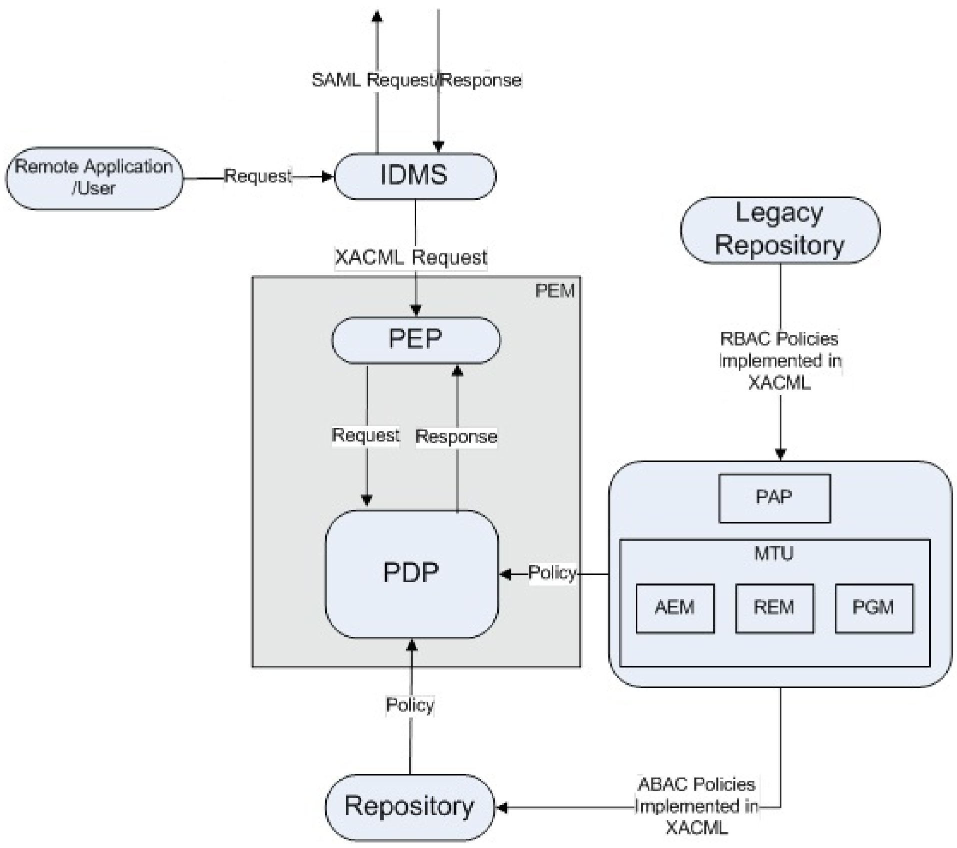


Figure 5.4: ABAC Plugin Architecture

5.4 Model Transformation Utility (MTU)

Model Transformation Utility reads RBAC policies form legacy repository and transforms it to ABAC model, using XACML and stores in ABAC policy store as shown in figure 5.4. These policies are applied to remote request to obtain access over local resources. When our proposed system is plugged-in to existing system it will read its entire legacy policy repository and generate ABAC policies implemented in XACML and store it into new repository. PDP while applying policy verify whether the given policy is up-to-date by matching its version with corresponding policy in legacy repository. If the given policy is out-dated it will first ask MTU to update it before applying it to given request. MTU comprised of three main modules:

- Attribute Extraction Module (AEM)
- Rule Extraction Module (REM)
- Policy Generation Module (PGM)

5.4.1 Attribute Extraction Module (AEM)

Attribute Extraction Module extracts attributes of subjects, resources and actions from the legacy RBAC model and store them into attribute repository. In RBAC model users are grouped into roles and permission are assigned to roles. Legacy system implemented RBAC model in Relational Database Management System. Entity relationship diagram of existing RBAC system is shown in Appendix A. RBAC user's attribute such as user's name and his role are mapped into ABAC subjects' attributes. AEM reads RBAC users and their corresponding roles attributes as shown in Table 5.2 and Table 5.3 and store them into ABAC attribute repository. RBAC operations are mapped to ABAC actions as shown in Table 5.4. Similarly, Table 5.5 shows extracted resources attributes of RBAC resources.

Id	**Name**	**Extracted Attributes in Repository**
1	Khalid	urn:lms:plugin:subject:names:name:Khalid
2	Imran	urn:lms:plugin:subject:names:Imran
3	Ali	urn:lms:plugin:subject:names:name:Imran

Table 5.2: Users

Id	Name	Extracted Attributes in Repository
1	Teacher	urn:lms:plugin:subject:roles:role:Teacher
2	Student	urn:lms:plugin:subject:roles:role:Student
3	Teaching Assistant	urn:lms:plugin:subject:roles:role:TeachingAssistant

Table 5.3: Roles

Id	Role(Name)	Extracted Attributes in Repository
1	Add	urn:lms:plugin:actions:name:Add
2	Edit	urn:lms:plugin:actions:name:Edit
3	Delete	urn:lms:plugin:actions:name:Delete
4	View	urn:lms:plugin:actions:name:View
5	Mark	urn:lms:plugin:actions:name:Mark
6	Publish	urn:lms:plugin:actions:name:Publish
7	Submit	urn:lms:plugin:actions:name:Submit
8	Download	urn:lms:plugin:actions:name:Download

Table 5.4: Operations

Id	Name	Extracted Attributes in Repository
1	Front Screen	urn:lms:plugin:resources:name:FrontScreen
2	Lecture Notes	urn:lms:plugin:resources:name:LectureNotes
3	Assignment	urn:lms:plugin:resources:name:Assignment
4	Course	urn:lms:plugin:resources:name:Course

Table 5.5: Resources

5.4.2 Rule Extraction Module (REM)

Rule extracting module is responsible of extracting rules from legacy RBAC relational database in order to generate ABAC policies. REM uses roles table, permissions table and their association tables for generating ABAC rules. REM first extracts set of allowable actions on a given resource by using resources table, operations table and their association table (Permissions) in relational database as shown in Table 5.6. Secondly, it uses RBAC roles table, permissions table and their association table in relational database to extracts ABAC rules as shown in Table 5.7.

Id	**Resource Id**	**Operation Id**	**ABAC Permission (Action on Resources)**
1	1	4	Front Screen can be viewed
2	2	1	Lecture Notes can be added
3	2	2	Lecture Notes can be edited
4	2	3	Lecture Notes can be deleted
5	2	4	Lecture Notes can be viewed
6	2	8	Lecture Notes can be downloaded
7	3	1	Assignment can be added
8	3	2	Assignment can be edited
9	3	3	Assignment can be deleted
10	3	4	Assignment can be viewed
11	3	5	Assignment can be marked
12	3	6	Assignment can be published
13	3	7	Assignment can be submitted
14	3	8	Assignment can be downloaded
15	4	1	Course can be downloaded
16	4	2	Course can be added
17	4	3	Course can be edited
18	4	4	Course can be deleted

Table 5.6: Permissions

Id	**Role Id**	**Permission Id**	**Extracted ABAC Rule**
1	1	1	User having Teacher role can view front screen
2	1	2	User having Teacher role can add Lecture Notes

Continued on next page

Table 5.7 – *Continued from previous page*

Id	Role Id	Permission Id	Extracted ABAC Rule
3	1	3	User having Teacher role can edit Lecture Notes
4	1	4	User having Teacher role can delete Lecture Notes
5	1	5	User having Teacher role can view Lecture Notes
6	1	6	User having Teacher role can download Lecture Notes
7	1	7	User having Teacher role can add Assignment
8	1	8	User having Teacher role can edit Assignment
9	1	9	User having Teacher role can delete Assignment
10	1	10	User having Teacher role can view Assignment
11	1	11	User having Teacher role can mark Assignment
12	1	12	User having Teacher role can publish Assignment
13	1	13	User having Teacher role can submit Assignment
14	1	14	User having Teacher role can download Assignment
15	1	15	User having Teacher role can download course
16	1	16	User having Teacher role can add course
17	1	17	User having Teacher role can edit course
18	2	1	User having Student role can view front screen
19	2	5	User having Student role view Lecture Notes
20	2	6	User having Student role download Lecture Notes

Continued on next page

Table 5.7 – *Continued from previous page*

Id	Role Id	Permission Id	Extracted ABAC Rule
21	2	13	User having Student role can submit Assignment
22	2	14	User having Student role can download Assignment
23	3	2	User having Teaching Assistant role can add Lecture Notes
24	3	3	User having Teaching Assistant role can edit Lecture Notes
25	3	11	User having Teaching Assistant role can mark Assignment
26	3	12	User having Teaching Assistant role publish Assignment

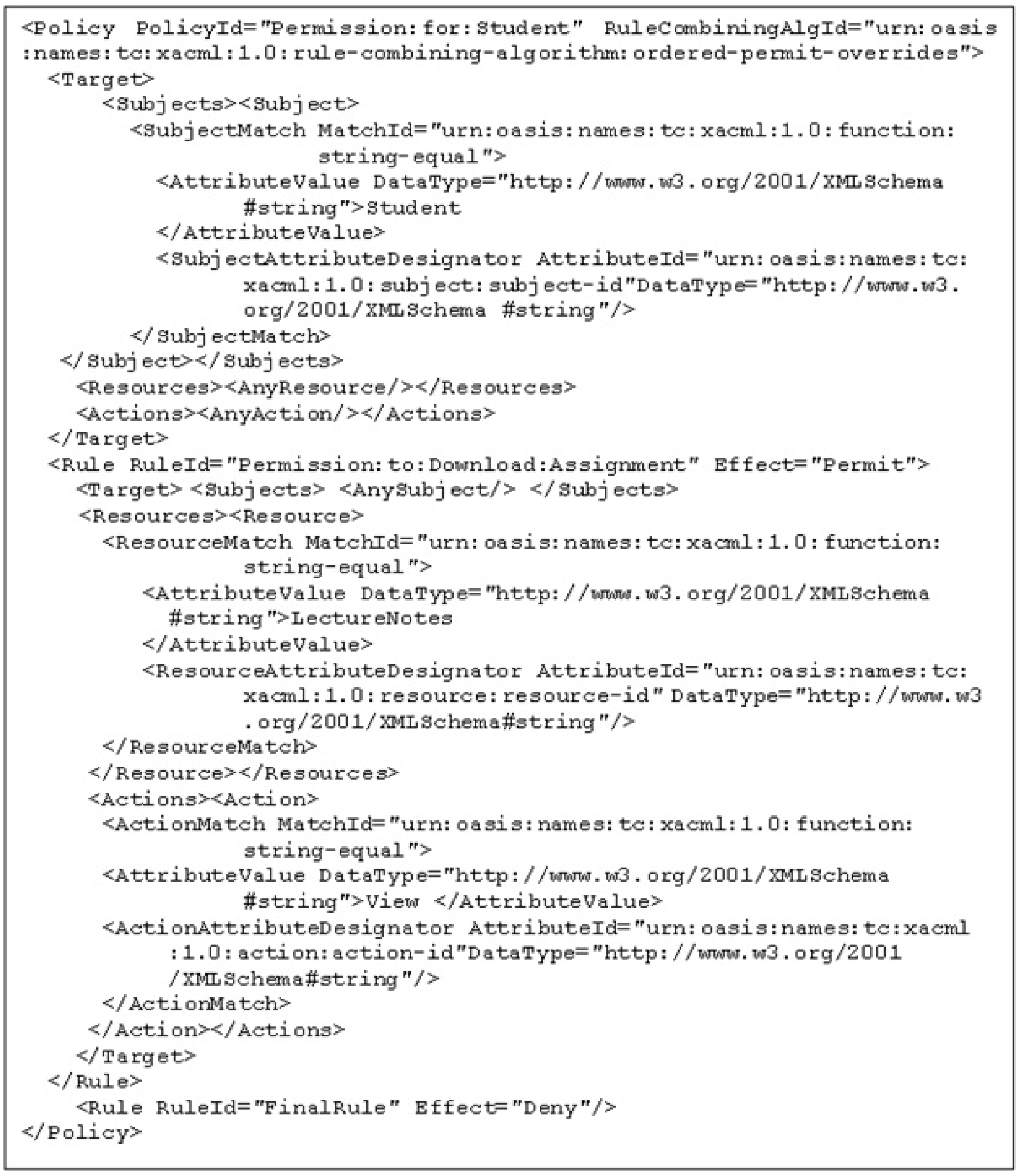
Table 5.7: Roles and Associated Permissions

5.4.3 Policy Generation Module (PGM)

Policy generation module is responsible for generating ABAC XACML policies based on attributes and rules extracted by AEM and REM respectively. PGM generates policy for each role defined in RBAC model. Figure 5.5 shows policy generated by PGM for Student role in which each rule corresponds to a single permission which student role have in RBAC model. ABAC policy target element contains condition on subject attribute in order to check that particular policy is applicable to given request or not. If user request has subject-id equal to "Student" then this policy is applied to it otherwise PDP looks for other applicable policies in ABAC policy store. Rules target in Student policy have no specific subject defined which mean it can be applied to any request for which policy is applicable. Resource and action attributes in rule target are compared with resource and action attributes in user's request respectively. If both values are matched rules returns "Permit" which mean user is permitted have requested access for the said resource otherwise next rule is checked. If no rule is applicable then rule with RuleId ="FinalRole" comes into play to deny access request.

5.5 Summary

ABAC plugin architecture includes two main modules: Policy Evaluation Module (PEM) and Model Transformation utility (MTU). MTU generates ABAC policies with the help of AEM, REM & PGM and PAP writes these policies into ABAC policy repository which are further used by PDP of PEM for decision making.



```
<Policy PolicyId="Permission:for:Student" RuleCombiningAlgId="urn:oasis
:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-overrides">
  <Target>
      <Subjects><Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                      string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">Student
          </AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                xacml:1.0:subject:subject-id"DataType="http://www.w3.
                org/2001/XMLSchema #string"/>
        </SubjectMatch>
   </Subject></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="Permission:to:Download:Assignment" Effect="Permit">
    <Target> <Subjects> <AnySubject/> </Subjects>
    <Resources><Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                string-equal">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
           #string">LectureNotes
         </AttributeValue>
         <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:
                xacml:1.0:resource:resource-id" DataType="http://www.w3
                .org/2001/XMLSchema#string"/>
      </ResourceMatch>
     </Resource></Resources>
     <Actions><Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">View </AttributeValue>
      <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml
           :1.0:action:action-id"DataType="http://www.w3.org/2001
           /XMLSchema#string"/>
      </ActionMatch>
     </Action></Actions>
    </Target>
  </Rule>
    <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

Figure 5.5: Policy (Student)

Chapter 6

Implementation and Results

In this chapter a functionality of each module with their corresponding graphical user interface is described. In the last section of this chapter we take an example to clarify authentication and authorization process.

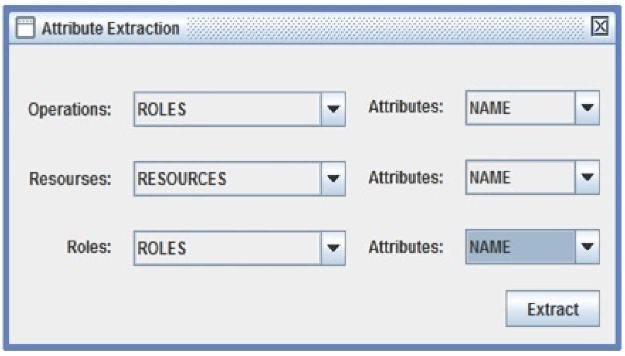
In order to understand the functionality of MTU we take university RBAC Model (discussed in previous section) and show how our MTU converts RBAC model into ABAC by first extracting attributes to build system vocabulary and then extracting rules. In the last step MTU generates XACML policies based on ABAC rules and attributes vocabulary.

6.1 Attribute Extraction

ABAC model is based on attributes. Graphical interface shown in figure 6.1 describes how we can use our system to extract attributes from existing RBAC model. User selects operations, resources and roles with their corresponding attribute (say name in our case) and click extract button to generate attribute file.
Figure 6.2 shows the snapshot of extracted attributes of resources (object), operations and roles in our existing RBAC model of typical university.

6.2 Rules Extraction

After successful generation of system vocabulary (ABAC attributes), MTU extract rules for ABAC policies. Rule extraction user interface is shown below in figure 6.3 in which user selects operations, resources, roles, permission (privileges), relationship between them (role_has_permission) and their



Operations: ROLES Attributes: NAME
Resourses: RESOURCES Attributes: NAME
Roles: ROLES Attributes: NAME

Figure 6.1: AEM User Interface

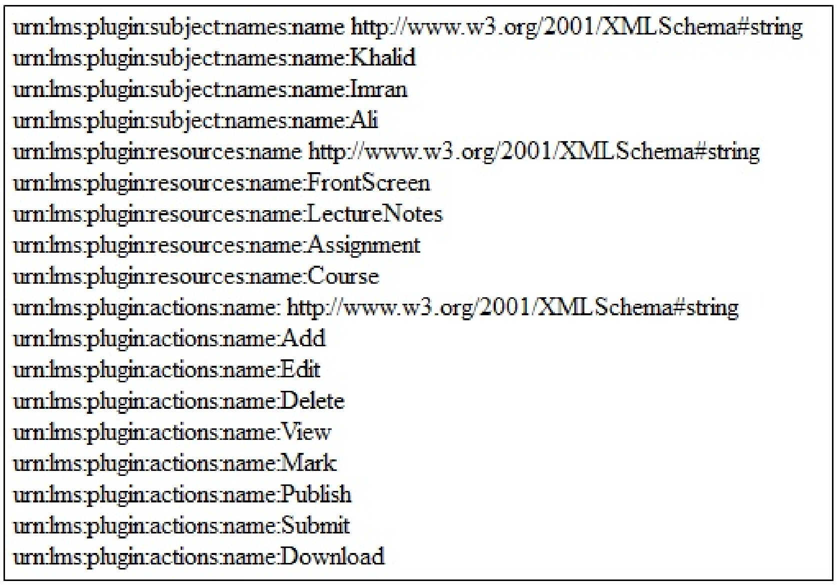
corresponding attributes. After selection user press extract and rules are generated. Rules extracted by REM are shown in figure 6.4.

6.3 Policy Generation

The last and the most important task of MTU is to generate ABAC policies in XACML. A permission policy is generated for every role defined in RBAC model. Each permission policy contains a set of rules, where each rule corresponds to ABAC rule generated by the rule generation module. In policy builder interface shown in figure 6.5 user selects attribute and rule files which were generated by attribute extraction module and rule extraction module in the last section and click generate button. As a result, a number of policy files are generated corresponding to roles listed in rule file. Each role in RBAC system corresponds to one policy in ABAC model implementation. Role policy (for student role) with set of rules, including default deny rule, is given in figure 6.6

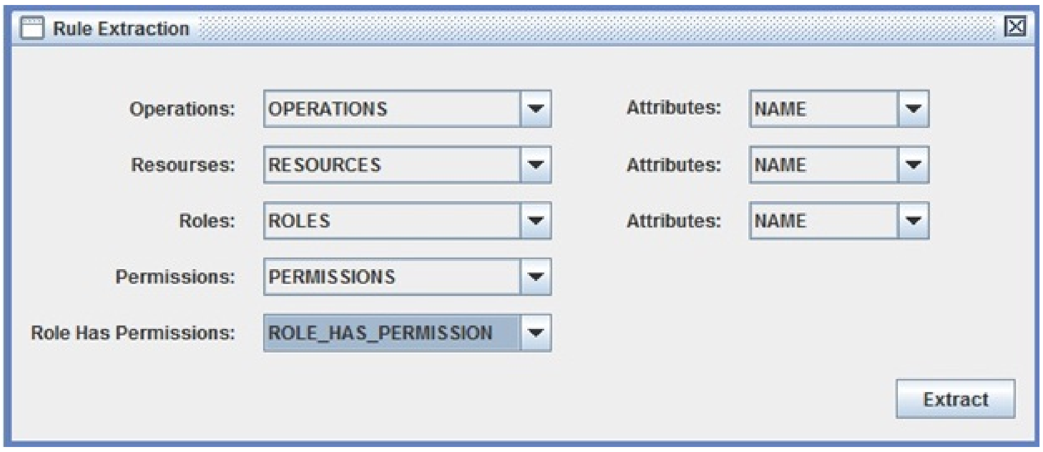
6.4 Authorization Cycle

In the previous section we transformed RBAC model into ABAC model and corresponding ABAC policies are generated in XACML and stored in repositories. In this section we will show how request from remote user (Software Company) will be processed and access is either granted or denied. Khalid (Trainee), being a user of Software Company wishes to view lecture notes and sends request to university learning management system. XACML request received at university IDMS system as shown below: XACML request has subject attribute value “khalid”, resource attribute value “Lecture Notes”



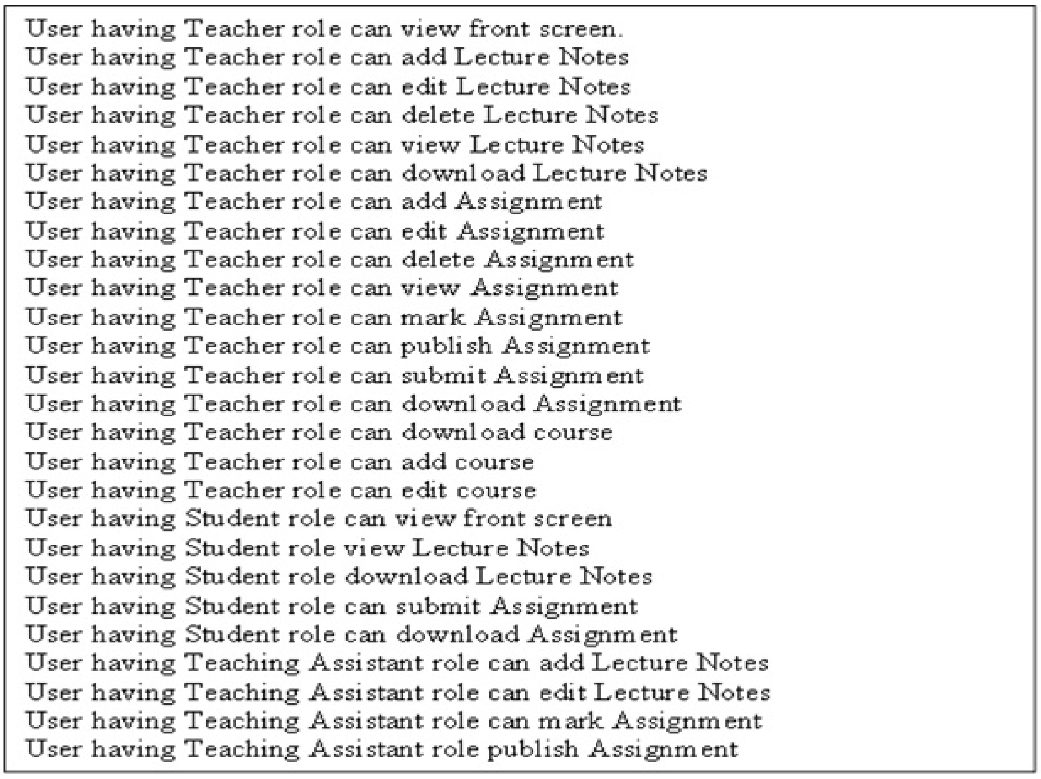
```
urn:lms:plugin:subject:names:name http://www.w3.org/2001/XMLSchema#string
urn:lms:plugin:subject:names:name:Khalid
urn:lms:plugin:subject:names:name:Imran
urn:lms:plugin:subject:names:name:Ali
urn:lms:plugin:resources:name http://www.w3.org/2001/XMLSchema#string
urn:lms:plugin:resources:name:FrontScreen
urn:lms:plugin:resources:name:LectureNotes
urn:lms:plugin:resources:name:Assignment
urn:lms:plugin:resources:name:Course
urn:lms:plugin:actions:name: http://www.w3.org/2001/XMLSchema#string
urn:lms:plugin:actions:name:Add
urn:lms:plugin:actions:name:Edit
urn:lms:plugin:actions:name:Delete
urn:lms:plugin:actions:name:View
urn:lms:plugin:actions:name:Mark
urn:lms:plugin:actions:name:Publish
urn:lms:plugin:actions:name:Submit
urn:lms:plugin:actions:name:Download
```

Figure 6.2: Attribute File



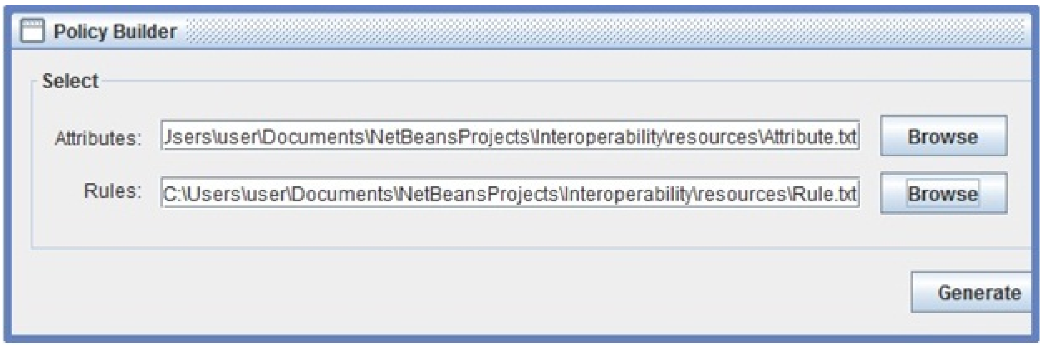
Operations: OPERATIONS — Attributes: NAME
Resourses: RESOURCES — Attributes: NAME
Roles: ROLES — Attributes: NAME
Permissions: PERMISSIONS
Role Has Permissions: ROLE_HAS_PERMISSION

Figure 6.3: REM User Interface



```
User having Teacher role can view front screen.
User having Teacher role can add Lecture Notes
User having Teacher role can edit Lecture Notes
User having Teacher role can delete Lecture Notes
User having Teacher role can view Lecture Notes
User having Teacher role can download Lecture Notes
User having Teacher role can add Assignment
User having Teacher role can edit Assignment
User having Teacher role can delete Assignment
User having Teacher role can view Assignment
User having Teacher role can mark Assignment
User having Teacher role can publish Assignment
User having Teacher role can submit Assignment
User having Teacher role can download Assignment
User having Teacher role can download course
User having Teacher role can add course
User having Teacher role can edit course
User having Student role can view front screen
User having Student role view Lecture Notes
User having Student role download Lecture Notes
User having Student role can submit Assignment
User having Student role can download Assignment
User having Teaching Assistant role can add Lecture Notes
User having Teaching Assistant role can edit Lecture Notes
User having Teaching Assistant role can mark Assignment
User having Teaching Assistant role publish Assignment
```

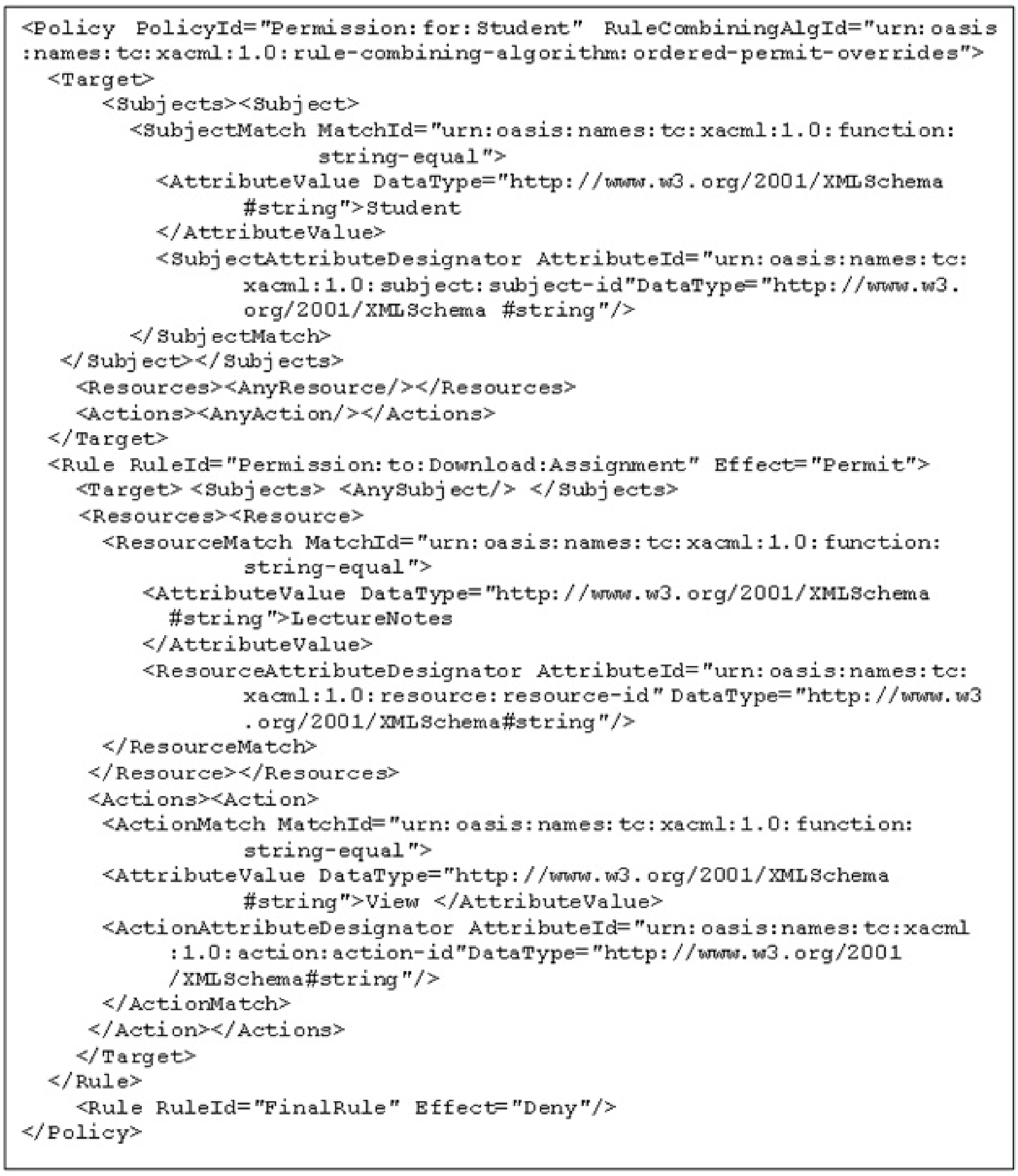
Figure 6.4: Rule File



Attributes: Jsers\user\Documents\NetBeansProjects\Interoperability\resources\Attribute.txt

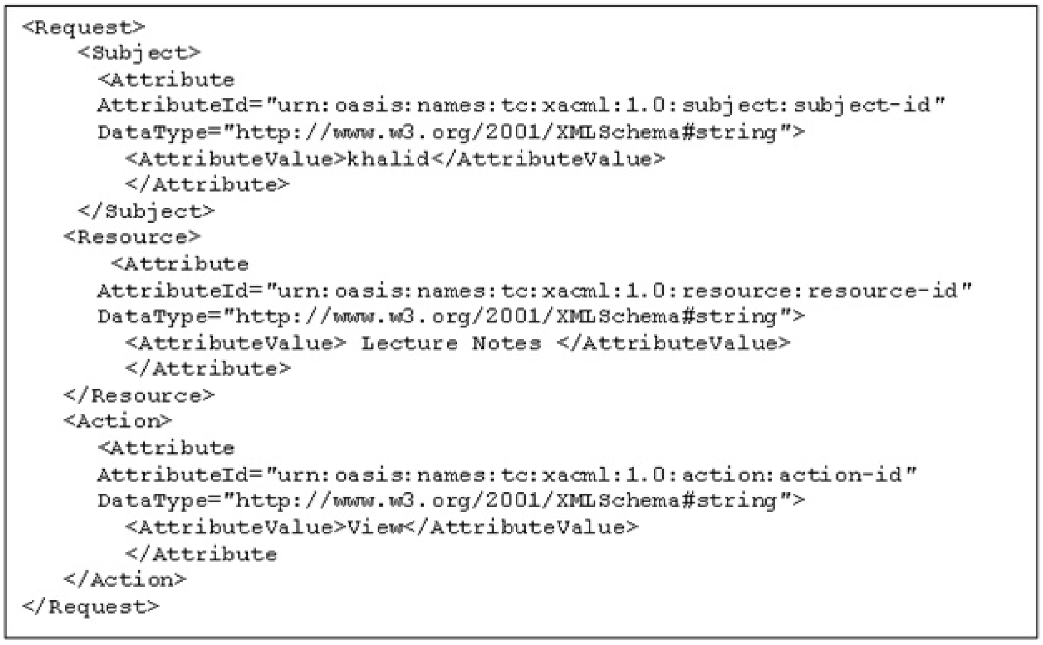
Rules: C:\Users\user\Documents\NetBeansProjects\Interoperability\resources\Rule.txt

Figure 6.5: MTU User Interface



```
<Policy PolicyId="Permission:for:Student" RuleCombiningAlgId="urn:oasis
:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-overrides">
  <Target>
      <Subjects><Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                      string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">Student
          </AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                xacml:1.0:subject:subject-id"DataType="http://www.w3.
                org/2001/XMLSchema #string"/>
        </SubjectMatch>
   </Subject></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="Permission:to:Download:Assignment" Effect="Permit">
    <Target> <Subjects> <AnySubject/> </Subjects>
    <Resources><Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                string-equal">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
           #string">LectureNotes
         </AttributeValue>
         <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:
                xacml:1.0:resource:resource-id" DataType="http://www.w3
                .org/2001/XMLSchema#string"/>
      </ResourceMatch>
     </Resource></Resources>
     <Actions><Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">View </AttributeValue>
      <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml
           :1.0:action:action-id"DataType="http://www.w3.org/2001
           /XMLSchema#string"/>
      </ActionMatch>
     </Action></Actions>
    </Target>
  </Rule>
    <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

Figure 6.6: Policy



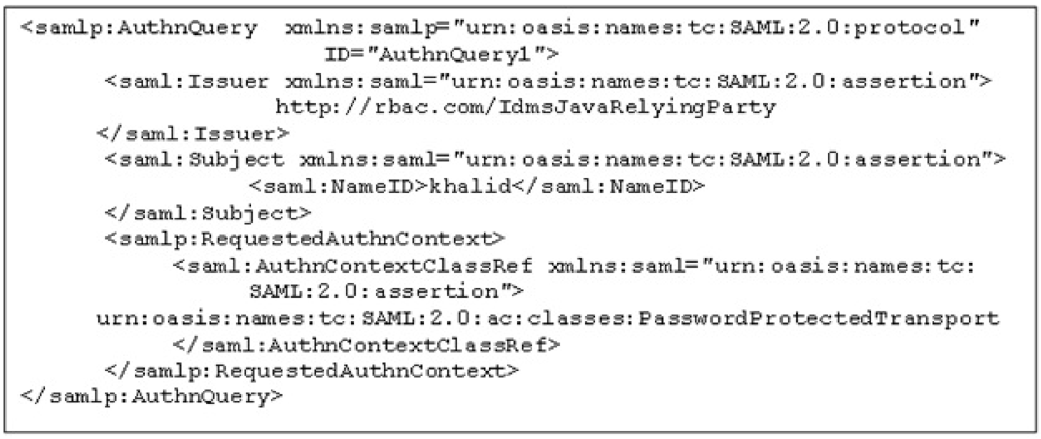
```
<Request>
    <Subject>
      <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>khalid</AttributeValue>
        </Attribute>
    </Subject>
   <Resource>
       <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue> Lecture Notes </AttributeValue>
        </Attribute>
   </Resource>
   <Action>
      <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>View</AttributeValue>
        </Attribute
   </Action>
</Request>
```

Figure 6.7: User Request

and action attribute value “View”. It states that a user (say khalid) want to view lecture notes.

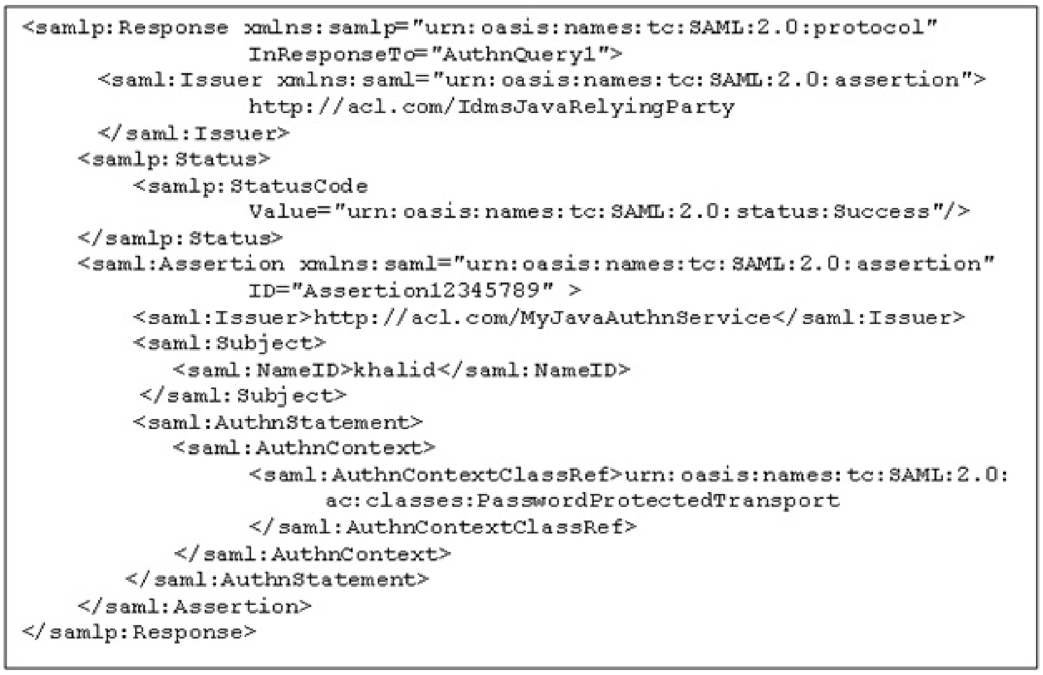
6.4.1 Authentication Request/Response

IDMS upon looking at the request from remote user, prepares SAML authentication request and forwards it to identity provider’s (software company) IDMS. User is authenticated in its identity provider’s IDMS and requested attributes are returned back to service provider (University). SAML authentication request for user (Khalid) and response with requested attribute (group) are shown in figures 6.8 and 6.9. After successful authentication as shown in SAML authentication query response, additional attribute such as “group” to which user belongs is requested from identity provider (IdP) by using SAML attribute query shown in figure 6.10. Figure 6.11 shows response of IdP’s (Software Company) SAML attribute query with group attribute value “Trainee” to service provider (University).
User interface for building request is shown in figure 6.12.



```
<samlp:AuthnQuery  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                      ID="AuthnQuery1">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                  http://rbac.com/IdmsJavaRelyingParty
     </saml:Issuer>
      <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                <saml:NameID>khalid</saml:NameID>
      </saml:Subject>
      <samlp:RequestedAuthnContext>
           <saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:
                SAML:2.0:assertion">
     urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
           </saml:AuthnContextClassRef>
      </samlp:RequestedAuthnContext>
</samlp:AuthnQuery>
```

Figure 6.8: SAML Authentication Query



```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                InResponseTo="AuthnQuery1">
     <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                http://acl.com/IdmsJavaRelyingParty
     </saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode
                Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
    <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                ID="Assertion12345789" >
        <saml:Issuer>http://acl.com/MyJavaAuthnService</saml:Issuer>
        <saml:Subject>
           <saml:NameID>khalid</saml:NameID>
         </saml:Subject>
        <saml:AuthnStatement>
           <saml:AuthnContext>
                <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:
                     ac:classes:PasswordProtectedTransport
                </saml:AuthnContextClassRef>
           </saml:AuthnContext>
        </saml:AuthnStatement>
    </saml:Assertion>
</samlp:Response>
```

Figure 6.9: SAML Authentication Query Response

```
<samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0:
                                  protocol" ID="AttrQuery2" >
      <saml:Issuer  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
           http://rbac.com/IdmsJavaRelyingParty
      </saml:Issuer>
      <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                <saml:NameID>khalid</saml:NameID>
      </saml:Subject>
      <saml:Attribute mlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
           Name="group"/>
</samlp:AttributeQuery>
```

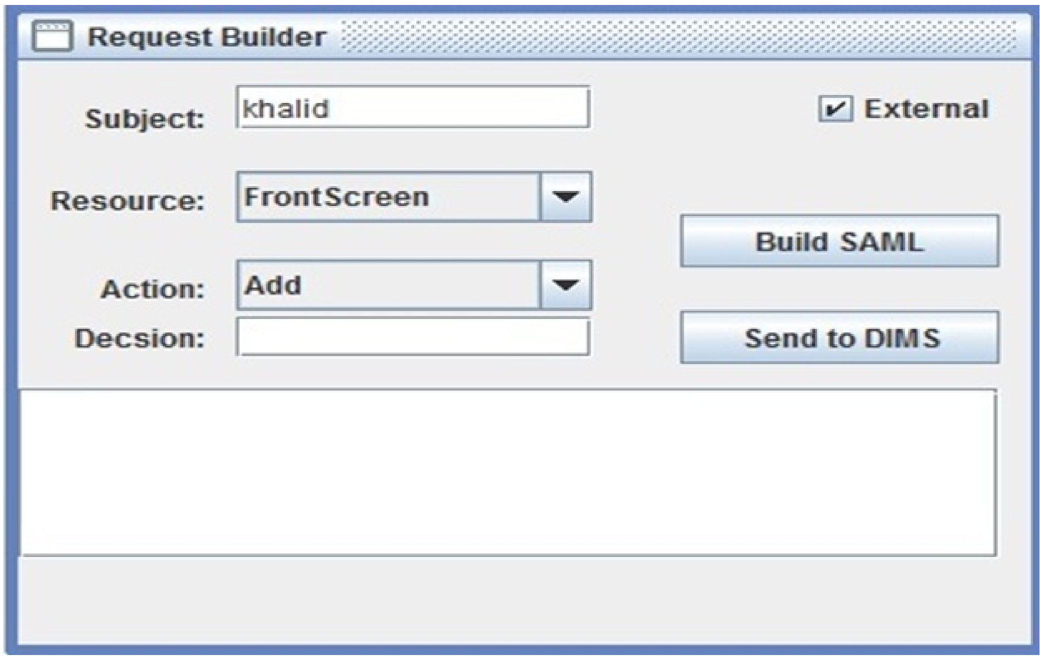
Figure 6.10: SAML Attribute Query

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                           InResponseTo="AttrQuery2">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
           http://acl.com/IdmsJavaRelyingParty
      </saml:Issuer>
     <saml:Assertion  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                                      ID="Assertion12345789">
      <saml:Issuer>http://acl.com/MyJavaAttributeService</saml:Issuer>
        <saml:Subject><saml:NameID Format="urn:oasis:names:tc:SAML:2.0:
                                      nameid-format:transient"> khalid
                    </saml:NameID>
        </saml:Subject>
        <saml:AttributeStatement>
           <saml:Attribute  Name="group">
                <saml:AttributeValue> Trainee </saml:AttributeValue>
           </saml:Attribute>
        </saml:AttributeStatement>
      </saml:Assertion>
</samlp:Response>
```

Figure 6.11: SAML Attribute Query Response

6.4.2 Request (to PEP)

After successful authentication IDMS consults with mapping entries, prepares request by replacing Trainee with Student and forwards it to PEP.PEP prepares request in native format and forwards to PDP that evaluates request according to ABAC policies written in XACML. Request to PEP is shown below in figure 6.13.


Request Builder

Subject: khalid

☑ External

Resource: FrontScreen

Action: Add

Decsion:

Figure 6.12: Request Builder User Interface

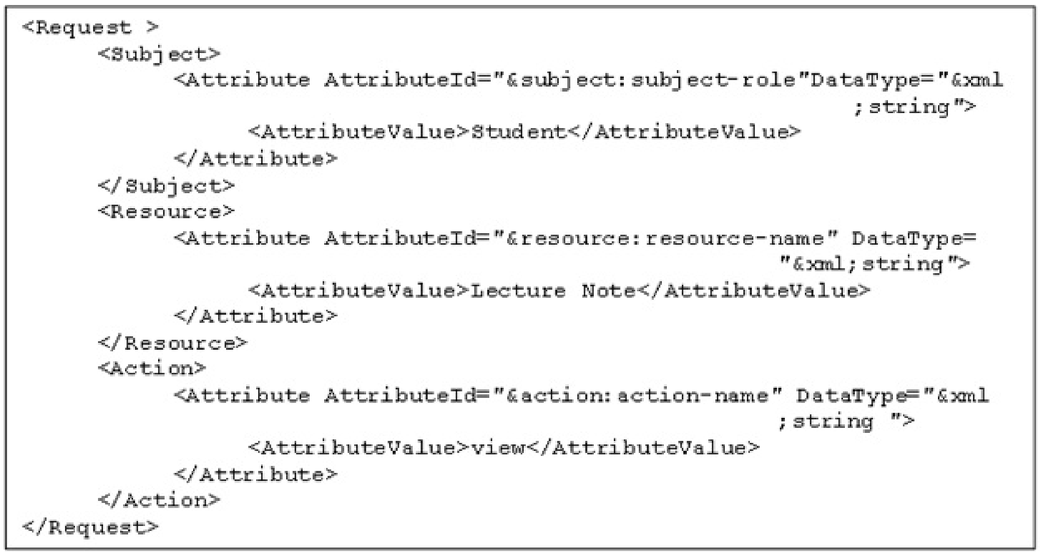

```
<Request >
     <Subject>
          <Attribute AttributeId="&subject:subject-role"DataType="&xml
                                                            ;string">
               <AttributeValue>Student</AttributeValue>
          </Attribute>
     </Subject>
     <Resource>
          <Attribute AttributeId="&resource:resource-name" DataType=
                                                       "&xml;string">
               <AttributeValue>Lecture Note</AttributeValue>
          </Attribute>
     </Resource>
     <Action>
          <Attribute AttributeId="&action:action-name" DataType="&xml
                                                       ;string ">
               <AttributeValue>view</AttributeValue>
          </Attribute>
     </Action>
</Request>
```

Figure 6.13: Request to PEP

6.5 Test Cases

In order check the correctness of transformation of RBAC model into ABAC XACML significant number of test cases are executed. Test cases and their outcomes are listed below:

6.5.1 Test Case 1

Test Case Title	Remote_Request_01
Test Case ID	TC_001
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Khalid) having “Teacher” role. **2.** Select “Front Screen” as resource. **3.** Select “View” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.2 Test Case 2

Test Case Title	Remote_Request_02
Test Case ID	TC_002
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having "Student" role. **2.** Select "Front Screen" as resource. **3.** Select "View" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.3 Test Case 3

Test Case Title	Remote_Request_03
Test Case ID	TC_003
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having "Student" role. **2.** Select "Lecture Notes" as resource. **3.** Select "View" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.4 Test Case 4

Test Case Title	Remote_Request_04
Test Case ID	TC_004
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having "Student" role. **2.** Select "Lecture Notes" as resource. **3.** Select "Add" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Denied
Actual Results	Denied
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.5 Test Case 5

Test Case Title	Remote_Request_05
Test Case ID	TC_005
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having “Student” role. **2.** Select “Lecture Notes” as resource. **3.** Select “Edit” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Denied
Actual Results	Denied
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.6 Test Case 6

Test Case Title	Remote_Request_05
Test Case ID	TC_006
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having "Student" role. **2.** Select "Lecture Notes" as resource. **3.** Select "Download" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.7 Test Case 7

Test Case Title	Remote_Request_07
Test Case ID	TC_007
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having "Student" role. **2.** Select "Lecture Notes" as resource. **3.** Select "Add" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Denied
Actual Results	Denied
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.8 Test Case 8

Test Case Title	Remote_Request_08
Test Case ID	TC_008
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having “Student” role. **2.** Select “Assignment” as resource. **3.** Select “Submit” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.9 Test Case 9

Test Case Title	Remote_Request_09
Test Case ID	TC_009
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having “Student” role. **2.** Select “Assignment” as resource. **3.** Select “Mark” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Denied
Actual Results	Denied
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.10 Test Case 10

Test Case Title	Remote_Request_10
Test Case ID	TC_010
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Imran) having “Student” role. **2.** Select “Assignment” as resource. **3.** Select “Publish” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Denied
Actual Results	Denied
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.11 Test Case 11

Test Case Title	Remote_Request_11
Test Case ID	TC_011
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Ali) having "Teaching Assistant" role. **2.** Select "Front Screen" as resource. **3.** Select "View" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.12 Test Case 12

Test Case Title	Remote_Request_12
Test Case ID	TC_012
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Ali) having “Teaching Assistant” role. **2.** Select “Lecture Notes” as resource. **3.** Select “Add” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.13 Test Case 13

Test Case Title	Remote_Request_13
Test Case ID	TC_013
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Ali) having "Teaching Assistant" role. **2.** Select "Assignment" as resource. **3.** Select "Publish" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.14 Test Case 14

Test Case Title	Remote_Request_14
Test Case ID	TC_014
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Ali) having “Teaching Assistant” role. **2.** Select “Assignment” as resource. **3.** Select “Mark” as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click “send to IDMS” button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.15 Test Case 15

Test Case Title	Remote_Request_15
Test Case ID	TC_015
Test Objective	To check the correctness of cross organization authentication and authorization process.
Pre Condition	User role and corresponding policies should exist.
Post Condition	User should be authenticated successfully and requested permissions should be granted.
Procedure	**1.** Enter subject (i.e. Ali) having "Teaching Assistant" role. **2.** Select "Lecture Notes" as resource. **3.** Select "Edit" as action. **4.** Check External checkbox. **5.** Click Build SAML Button. **6.** Click "send to IDMS" button.
Expected Results	Permit
Actual Results	Permit
Status	Pass
Carried Out on	1 August,2012
Carried Out by	Khalid Hafeez

6.5.16 Results

A number of test case scenarios have been generated and executed on both existing system and our transformed system to compare results. Same type of request is send to both existing and new transformed models implemented system. Result summarized in table below shows our transformation of legacy RBAC model into ABAC Model is 100% correct.

6.6 Summary

AEM and REM extracts RBAC system attributes and authorization rules respectively and stores them into file. PGM reads attribute and rules from files and generate ABAC policies. These ABAC policies stored in ABAC plugin repository are used evaluate remote request.

SAML assertion are used authenticate users coming from remote organization before access is provided by service provider.

In order to check the correctness of system, a number of test cases are generated and executed on both existing and transformed system and results are compared which shows that model transformation is 100% correct.

Sr.	**Request**	**Expected Result**	**Actual Result**
1.	User with teacher role wants to View Front Screen	Permit	Permit
2.	User with student role wants to View Front Screen	Permit	Permit
3.	User with student role wants to View Lecture Notes	Permit	Permit
4.	User with student role wants to Add Lecture Notes	Denied	Denied
5.	User with student role wants to Edit Lecture Notes	Denied	Denied
6.	User with student role wants to Download Lecture Notes	Permit	Permit
7.	User with student role wants to Add Lecture Notes	Denied	Denied
8.	User with student role wants to Submit Assignment	Permit	Permit
9.	User with student role wants to Mark Assignment	Denied	Denied
10.	User with student role wants to Publish Assignment	Denied	Denied
11.	User with teaching assistant role wants to View Front Screen	Permit	Permit
12.	User with teaching assistant role wants to Add Lecture Notes	Permit	Permit
13.	User with teaching assistant role wants to Publish Assignment	Permit	Permit
14.	User with teaching assistant role wants to Mark Assignment	Permit	Permit
15.	User with teaching assistant role wants to Edit Lecture Notes	Permit	Permit

Table 6.1: Results

Chapter 7

Conclusion & Future Work

7.1 Conclusion

In this research, we provide a decentralized and distributed solution by making access control interoperable, for federated organizations (implemented different access control models) collaborating with each other and accessing resources mutually by authenticating/authorizing each other's users seamlessly.

Enterprise distributed yet collaborative applications rapidly raise the need of comprehensive distributed security frameworks. In order to provide seamless authentication throughout the enterprise and among application of federated, collaborative, traditional centralized solution suffers with single point of failure. Centralized server may be overloaded when more than one federation is involved. Shibboleth centralized WAYF service for single sign on, inherits single point of failure problem for being centralized in nature. Other RBAC family distributed access control frameworks (TBAC, TMAC and CBAC) are enhancements of core RBAC model. Each of the above models requires major infrastructure changes for organizations subject to implementation.

We have proposed a distributed solution for collaborating organization in order to control access to their resources by making access control model interoperable. MTU transforms the policies written in native access control model into ABAC policies, implemented in XACML. We give a proof of concept by transforming RBAC policies to ABAC policies showing that how interoperability can be helpful for communication and resource sharing among collaborative organizations and in between enterprise modules. Our plug-in approach does not interfere with the internal workflow of both collaborating organizations thus requires no infrastructure changes. The advantage of this solution is that we need to write MTU as a part of plugin for each new access control model for being interoperable with other models employed in collaborative federations.

7.2 Future Work

Access control modeling is a vast domain and still has a number of challenges that need to be addressed in order to provide a foolproof security mechanism. In order to address the challenges of above mentioned domain, we have recommended the following future directions in which research can be accomplished.

- This work gives the notion of achieving interoperability among access control model by transforming them into an intermediate model (Attribute Based Access Control Model). Both collaborating organization's existing access control models are transformed into ABAC by Model transformation utility, automatically. Transactions across the organization are handled by new ABAC model, where existing authentication and authorization flows are untouched.

- In this research, we transform core RBAC model into ABAC model. MTU can be further enhanced to provide support for transformation of hierarchal and statically/dynamically constrained RBAC model.

- In future, we can add support to other models such as Access Control List (ACL), Purpose Based Access Control Model (PBAC), Usage Control Model (UCON), Temporal Based Access Control model (TBAC) etc. by writing model transformation utility for each of the models in order to make them interoperable with rest of the transformed models.

Bibliography

Bell, D. E. and La Padula, L. J. (1976). Secure computer system: Unified exposition and multics interpretation. Technical report, DTIC Document.

Biba, K. (1977). Integrity considerations for secure computer systems. Technical report, Technical Report MTR-3153.

Cohen, E., Thomas, R. K., Winsborough, W., and Shands, D. (2002). Models for coalition-based access control (cbac). In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 97–106. ACM.

Daniel (2005). Differences between xacml versions 1.0 and 2.0. `http://xml.coverpages.org/JoslinXACMLDiffs1-2.html`.

Erik Rissanen, A. A. (2010). xacml-3.0-core-spec-cs-01-en. `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf`.

Hu, V. C., Ferraiolo, D., and Kuhn, D. R. (2006). *Assessment of access control systems*. US Department of Commerce, National Institute of Standards and Technology.

Lepro, R. (2004). Cardea: Dynamic access control in distributed systems. *System*, 13(13):4–2.

Li, N., Hwang, J., and Xie, T. (2008). Multiple-implementation testing for xacml implementations. In *Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications*, pages 27–33. ACM.

Liu, A. X., Chen, F., Hwang, J., and Xie, T. (2008). Xengine: a fast and scalable xacml policy evaluation engine. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 265–276. ACM.

Lorch, M., Proctor, S., Lepro, R., Kafura, D., and Shah, S. (2003). First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, pages 25–37. ACM.

Lorch, M., Ribbens, C. J., Ramakrishnan, N., Varadarajan, S., and Hicks, J. O. (2004). Prima-privilege management and authorization in grid computing environments.

Ltd, H. V. (2006). Single sign on. `http://www.authenticationworld.com/Single-Sign-On-Authentication/`.

Microsystems, S. (2004). Sun's xacml implementation programmer's guide. `http://sunxacml.sourceforge.net/guide.html`.

OASIS (2005a). access-control-xacml-2.0-policy-schema-os. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-policy-schema-os.xsd`.

OASIS (2005b). Differences between xacml 2.0 and xacml 3.0. `http://xml.coverpages.org/JoslinXACMLDiffs1-2.html`.

OASIS (2005c). Saml specifications. `http://saml.xml.org/saml-specifications`.

OASIS (2010). xacml-core-v3-schema-wd-17. `http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd`.

OASIS (2013). Oasis- advancing open standards for the information security. `http://www.oasis-open.org/standards`.

Patil, V., Mei, A., and Mancini, L. V. (2007). Addressing interoperability issues in access control models. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 389–391. ACM.

Provost, W. (2009). Opensaml examples. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf`.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Scott Cantor, Brent Putman, R. W. I. Y. T. Z. T. Z. (2003). Shibboleth- a project of the internet2 middleware initiative. `http://shibboleth.net/`.

Simon Godik, T. M. (2003). cs-xacml-specification-1.1. `http://docs.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf`.

Thomas, R. K. (1997). Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, pages 13–19. ACM.

Thomas, R. K. and Sandhu, R. S. (1998). Task-based authorization controls (tbac): A family of models for active and enterprise-oriented authorization management. *Database Security*, 11:166–181.

Tim Moses, E. I. (2005). oasis-access-control-xacml-2.0-core-spec-os. `http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-04.pdf`.

Vincent C. Hu, David F. Ferraiolo, D. R. K. (2006). A survey of access control models. `http://csrc.nist.gov/news_events/privilege-management-workshop/PvM-Model-Survey-Aug26-2009`.

`pdf.`

Zhu, J. and Smari, W. W. (2008). Attribute based access control and security for collaboration environments. In *Aerospace and Electronics Conference, 2008. NAECON 2008. IEEE National*, pages 31–35. IEEE.

Appendix A

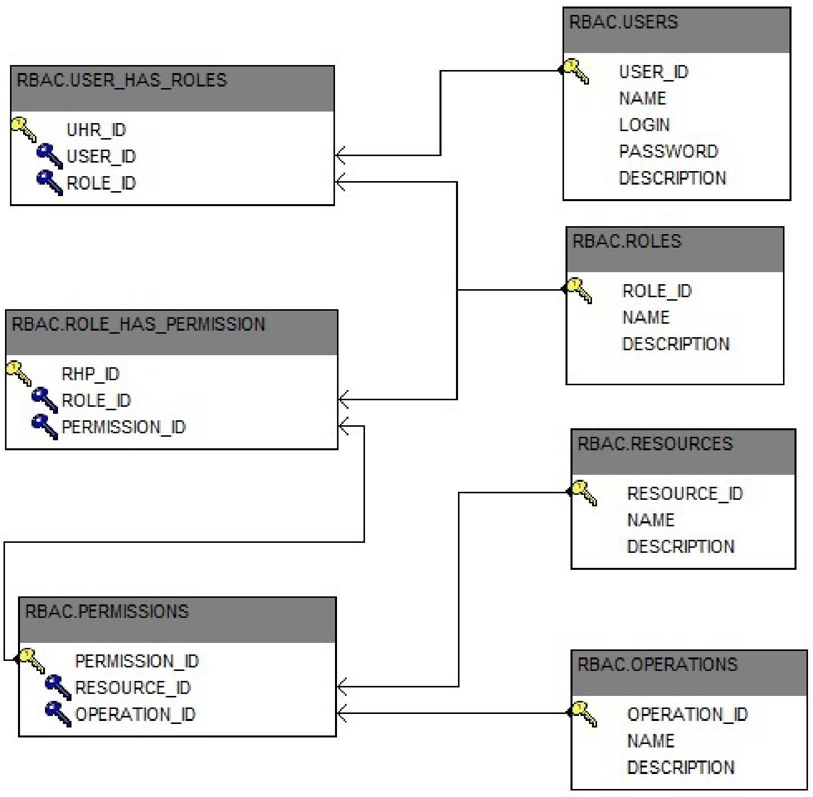


Figure 7.1: RBAC Relational Model