A
Dissertation
On

Representing Access Control Policies in OWL

Submitted in Partial Fulfilment of the Requirement

For the Award of the Degree of

Master of Technology

In

Computer Science & Engineering

Submitted By
Varsha Rani Sharma
2K14/CSE/26

Under the Esteemed Guidance of
Mr. Manoj Kumar
(Associate Professor)

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

JUNE 2016

Department of Computer Engineering

Delhi Technological University

Delhi-110042

CERTIFICATE

This is to certify that the dissertation titled “**Representing Access Control Policies in OWL**” is a bonafide record of work done at **Department of Computer Engineering, Delhi Technological University** by **Ms. VARSHA RANI SHARMA, Roll No. 2K14/CSE/26**, in partial fulfilment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering.

This project work was carried out by her under my guidance and supervision. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma in any university/institution to the best of my knowledge and belief. Her work is found to be outstanding and her discipline impeccable during the course of the project.

I wish her success in all her endeavours.

(Mr. Manoj Kumar)
Associate Professor and Project Guide
Department of Computer Engineering
Delhi Technological University

Date: __________

ACKNOWLEDGEMENT

First of all, let me thank the almighty god, my parents and my dear friends who are the most graceful and merciful for their blessing that contributed to the successful completion of this project.

I would like to devote my gratitude and thanks to my guide Sh. Manoj Kumar, Associate Professor, Department of Computer Engineering, Delhi Technological University, Delhi for his valuable guidance, constant encouragement and helpful discussions throughout the course of this work. Obviously, the progress I had now will be uncertain without his guidance.

I would also like to thank Prof. (Dr.) O.P. Verma, H.O.D. Computer Engineering Department, Delhi Technological University, Delhi for providing me better facilities and constant encouragement.

I would like to take this opportunity to express the profound sense of grati- tude and respect to all those who helped us throughout the duration of this project. DELHI TECHNOLOGICAL UNIVERSITY, in particular has been the source of inspiration, I acknowledge the effort of those who have contributed significantly to this project.

Varsha Rani Sharma
University Roll No.: 2K14/CSE/26
MTech (Computer Science and Engineering)
Department of Computer Engineering
Delhi Technological University

ABSTRACT

Organizations need access control to restrict the use of information related to them. Contextual parameters play a key role to control the access to the information. However, classical access control models do not have an explicit way to include context in access control. In this thesis, we propose an extension to the existing Role Based Access Control (RBAC) system where the context parameters can also be included. We have modelled context information like time, day, location, group membership etc. The proposed framework is extensible enough to add more contextual parameters as per the need.

Access control, in organizations, is driven by policies captured according to the access control model in use. There is always a need to have an automatic and adaptive access control system. In this work, we propose the representation of policies and the access control model in Web Ontology Language (OWL). This representation provides a formal way to achieve automation. We enforce these policies by making use of an inference based reasoner. This process is based on deducing additional facts from given data and leverages the semantic nature of OWL. We use this information, collectively, in making access control decisions.

We also show that the proposed framework can be used in many real world organizations by demonstrating its application to academic domain. Ontologies have been written to capture different aspect of the academic system including roles, permissions and contextual parameters. As a specific example, an access control system for the examination portal at DTU has been designed and developed. This system shows how access to different webpages is governed by different contexts. The system also provides features like adding new policies and modifying existing ones. The developed system shows the potential capability of our proposed framework and can be extended to other applications as well.

Contents

List of Figures

© 2016, Delhi Technological University, Delhi

List of Tables

Chapter 1

Introduction

1.1 Motivation

Access control is a one of the important issues in information security. Access control is used to control the flow of information which is related to the organization. Access control systems determine that if access to resources are allowed or not. And if the access is allowed then what are the criteria for that. The process of access control is governed by an underlying access control model and authorization policies. The policy dictates the intent of the organization as to what is permitted under which situations.

There are many access control models available for the organization to choose. However, every model comes with its own advantages and disadvantages. While selecting the access control models, organization analyze how much flexibility is provided by that model. They also consider what happens to the model when the operating environment changes. This issue deals with the extensibility of the access control model.

There are mainly three access control models which are quite popular. These are Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC). Although we cover these model in detail in chapter 2, we give some salient features of these models to explain the motivation behind our work.

DAC restricts access to resources based on the identity of subjects while

MAC provide access control based on the security levels associated with subjects and resources or objects. In RBAC, access permissions are encapsulated in roles. Roles are then assigned to subjects or users. The problem with DAC is that resources or information may be accessed by users which are not authorized because it does not provide a way to control the access of multiple copies of the same object. MAC deals with this problem by associating different security levels with subjects and objects. However, DAC and MAC do not provide flexible access control because policies in these models are fixed.

RBAC addresses these issues by proposing the concept of roles. Objects can only be accessed by those subjects who have compatible roles with respect to that object. This compatibility is decided by the system administrator and guided by the authorization policy of the organization. RBAC has become the most prevalent access control model in practice today. However, RBAC suffers with the problem of role explosion. It is due to the fact that organization may end up having too many different roles to handle different types of situations. The problem becomes very difficult when contextual information like `day`, `time` *etc.* need to be incorporated in the access control decision.

In order to cover access control for practical situations, we need an automatic way of access control. For achieving this, we need a formal way to represent policy and access control model that can be understood by the machine. The Web Ontology Language (OWL) is a policy language which can be used for writing authorization policies and defining access control models. Policies written in OWL can be reused across different organisations of different domains. Organizations also want to combine authorization policies and access control model and enforce access control based on inference based reasoning process.

All these factors clearly state the requirement to design and develop an access control framework where policies and access control models can be combined and used for automated policy enforcement.

1.2 Objective and Research Challenges

The main aim of this work is to represent access control model based on roles and associate contextual information with them. We need to come up with a framework which should be flexible and extensible enough to add more features as and when the need arise. Further, the representation is required to be in a machine understandable format for automatic enforcement. Traditional access control models are very rigid in nature and does not provide a flexible and adaptable framework. It is difficult to represent context information like from where the request is made or at what time or on what day request is made with traditional access control systems. The scope of our work includes addressing these issues and adding the required flexibility to provide better access control options in organizations.

There are some associated challenges with the design and development of this access control framework. The first one is that organisations have different policies which tells the intent of an organisation. These policies are generally written in higher level language. We need a way to represent these policies such that they can be understood by machines so that access control can be automatized. The next one is how contextual information would be added and how much flexibility can be provided.

Next issue deals with the fact that most organizations already have RBAC system in place. The challenge is to integrate our framework with their existing system seamlessly. Finally, the performance of the reasoning process is an important issue. The time taken should be acceptable to most of the practical situations.

Hence we can say that the objective of this work is to represent access control policies in OWL. The representation must be flexible enough and contextual information should be included in the model representation. The access control system should provide automatic access control and should also be reusable or customizable across different domains.

1.3 Contribution

The project work was carried out as per the set objectives. We have addressed the key research issues mentioned in the previous section. The contributions made towards the project work are summarized below:

1. **Representing Policies and Access Control Model**: We have represented the Role Based Access Control (RBAC) for academic domain. Ontologies have been written to represent various roles and the relationship among them. The instantiation has been done in data ontology. We have written various policy rules captured by underlying RBAC model. The language used for the representation was OWL. Although OWL is a language to represent the we information, it has been successfully used to represent security policies.

2. **Incorporate Contextual Information**: The RBAC model has been extended to incorporate various contextual information. Examples include `day`, `time`, `location` *etc.*. The extended framework is flexible and many more contextual information can be added as per the need.

3. **Enforcement based on Reasoning**: The combination of contextual RBAC and authorization policy is subjected to a reasoning framework. The nature of OWL is semantic which means additional facts can be inferred from the given data facts. This feature helps us to reason about the access request and decide what is permitted. We have used two different reasoners in our work, *viz.* EYE [2] and Cwm [3]. EYE is proven to outperform other reasoners for all practical purpose [4].

4. **Demonstration of a Working System**: We have demonstrated the capability of our representation framework by implementing an access control system for the examination portal of DTU. The developed access control system shows how real time restriction can be imposed on different types of information. This also shows how different kind of contextual information can be incorporated while making the access control decisions.

1.4 Outline

This thesis work is henceforth organized as follows. Chapter 2 deals with basic overview of the access control concept, issues and challenges in current access control systems and over view of policy language used. Chapter 3 provides representation of proposed access control model and representation of ontologies for the current working domain defined in OWL.

Chapter 4 Provides implementation details of the developed access control system for the examination portal at Delhi Technological University (DTU). This chapter also covers the policy administration part. Finally, we conclude in chapter 5 with brief comments on our future work.

Chapter 2

Literature Review

Organizations need an automatic access control system where policies can be defined and enforced based on varying context information. There has been considerable efforts in this area in last few years. We now present the related work which has been done in this area. The related work in the area can be catagorized into three parts: *access control models*, the *access control policies specification* and *reasoning based policy enforcement.* The need to design and develop a distributed, dynamic access control system where authorization is decided dynamically at run-time, is achieved by combining all these parts using semantic logic. We now discuss all these parts, in detail, based on the extensive literature review.

2.1 Access Control Models

Access control is a way to control or restrict the access to resources. Access control systems are used in organizations to control the flow of information related to the organisation. This is guided by authorization policies which represent the intent of the organization. Usually different organisations have different policies. These policies are captured using one of the access control model. There are three most popular access control models which are used in practice. These are Discretionary Access Control(DAC) [5], Mandatory Access Control (MAC) [6] and Role Based Access Control (RBAC) [1].

2.1.1 Discretionary Access Control (DAC)

In discretionary access control, permissions to access resources are assigned on the basis of user identity. The identity of the user decides authorization associated with each resource in the system. In this type of access control each object in the system has associated allowed access modes for each user based on the identity. For example, a user `Bob` is allowed to `read` and `write` a file but not allowed to `execute` that. Generally this is specified using the data structure called as *access control list* for each object. In this list the identity of each user is specified with the access modes that she has on that object. Each access request for a particular resource is checked against the specified authorization associated with that resource and it is determined if the user is allowed to access this resource in specific mode. If so, then access is granted otherwise access is denied.

One major drawback of DAC is that this model does not specify the control on the flow of information through the system in case of multiple copies of an information object. For example, if a user makes copy of a file which he can read then associated authorization information will be lost in the new copy of the resource. Any other user which is not authorized to access that resource in original, can access the new copy of that resource. Other drawback is that if a user leaves the organization then all entries of that user associated with all object in the system must be removed which is a tedious task in large organisations.

2.1.2 Mandatory Access Control (MAC)

In mandatory access control, each subject and object in the system are assigned security levels with them. The security level for a subject is known as its *clearance* and the level associated with an object is known as *classification.*

Permission to access particular object is granted if there exists some consistent relationship between the *clearance* of the subject and the *classification* of the object. This is generally specified by some multi-level security model. For example, Bell LaPadula [7] security model assigns four security levels. These levels are specified, in decreasing order of their significance, as:

`Top Secret`, `Secret`, `Confidential` and `Unrestricted`. MAC is generally used in military arenas where information is classified into multiple levels.

According to the Bell Lapadula [7] model, the access to objects are allowed based on two simple rules: *Read Down* and *Write Up*. The first rule says that subjects can read only those object which have a *classification* level equal to lower than the *clearance* of the subject. The second rule says that for writing an object, the *clearance* level of the subject must be lower than the *classification* level of the object.

Although MAC solves the problem of DAC as the security levels also apply on the newly created copy, it has the drawback that it is very rigid in the sense that information is classified into very few levels. Because of this, mandatory access control is not really useful for commercial organisation where information can not be classified in few levels .

2.1.3 Role Based Access Control (RBAC)

Role based access control overcome the limitations of discretionary access control and mandatory access control by introducing the concept of *roles*. In RBAC, access permissions are granted on the basis of the activated roles of the users which they have in an organisation. Role based access control is flexible as in this model permissions are assigned to the roles and roles are further assigned to users or subjects. In RBAC, subjects are the entities which makes the request to access particular resource on behalf of user.

In order to design a RBAC system, *roles* are identified on the basis of activities in the organization which are supposed to be performed. *Roles* are then assigned to users based on the responsibilities of the users which they bear as a part of system. Permissions are then assigned to *roles*. A user can perform all actions which are permitted to perform by her role. In some systems users can have multiple roles and one role can be assigned to multiple users.

It is easy to manage authoriaztion using RBAC system. Authorization management is required, for example, when a user leaves the organization or the responsibilities of an existing user change. In order to manage the change of responsibility case, the only thing which is required is to reassign the required *role* to the user and deactivate some of the previously assigned *roles*, if required. The case when the user leaves the organization is handled by assigning her *role* to some other user to carry out her duties. This is a huge advantage as only few change would be required in RBAC as contrast to DAC in which major changes must be done in the access control list of every objects in the system.

The RBAC model adopted by NIST [1] is divided into four levels of increasing operational capabilities and called as flat RBAC, hierarchical RBAC, constrained RBAC and symmetric RBAC. These levels are progressive and each adds exactly one new functionality. The basic structure of RBAC controls access of user to system based on the activities of user which he can perform. Roles are identified based on the activities in the system and assigned permissions. After this, users are associated with roles and get the permissions conjoined with the roles. Users can activate roles from one of the associated roles in any sessions. A request made by a user to perform an action is permitted if user's currently activated role is allowed to perform that action. Advanced RBAC features are role hierarchy and constraints. In next section we discuss the various RBAC forms in detail:

2.1.3.1 Flat RBAC

The Flat RBAC is the basic version of RBAC and covers only the essential features of RBAC. The main idea behind RBAC is that subjects (or users) are associated with roles, permissions are assigned to roles, and subjects (or users) get rights to access an object permissions based on the role they hold. The NIST RBAC [1] states that relationship between user and role can be many-to-many. The same is the case for relationship between permission and role assignment. This means that one single user can be assigned to many roles and a single role can have many users. Flat RBAC poses the requirement that roles assigned to a particular user and users assigned to a particular role should be determined. Flat RBAC also requires that users should be able to simultaneously hold permissions assigned to multiple roles.

2.1.3.2 Hierarchical RBAC

Hierarchical RBAC introduces the concept of role hierarchies and provide a support for that. Role hierarchy is a natural way in organizations to distribute the responsibilities among its users. This directly corresponds to the type of job function associated with the user. For example, in an academic institute, `student` is identified as a role. However, there are different types of students based on the course they do. This may include `UG Student`, `PG Student`. The `PG Student` can be further subdivided to `MTech Student` and `Phd Student` and so on.

The role hierarchy states that the action which is allowed for a higher level role user, is also allowed for a specific subdivision of that role. For example, if `student` is allowed to access library printer then it implies that all types of students down in the hierarchy are allowed to do so. This allows for more flexibility in the system for managing roles and authorizations associated with them.

2.1.3.3 Constrained RBAC

Constrained RBAC introduces the concept of separation of duties constraints. These constraints can be of two types: *static separation of duties* and *dynamic separation of duties.*

The *static separation of duties* constraint says that the user cannot hold two different roles. For example, the roles `student` and `librarian` have static separation of duties between them. A user cannot have these two possible roles. The *dynamic separation of duties* constraints, on the other hand, dictates that user can have these roles but they cannot be activated simultaneously in a session at the same time. The example is `loan manager` and `customer`. Although, the loan manager can also apply for loan but he would not be the sanctioning authority for the same. So if the `customer` role is activated then `loan manger` role cannot be activated in that session.

Figure 2.1 shows the the concept of Static Separation of Duties (SSD) in role hierarchy. The Permission Role Management System (PRMS) keeps track of which operations (OPS) are allowed on which objects (OBS). This is managed through different roles defined in the system. The Dynamic Separation of Duties (DSD) would be applicable for session roles which has not been shown explicitly in the figure.

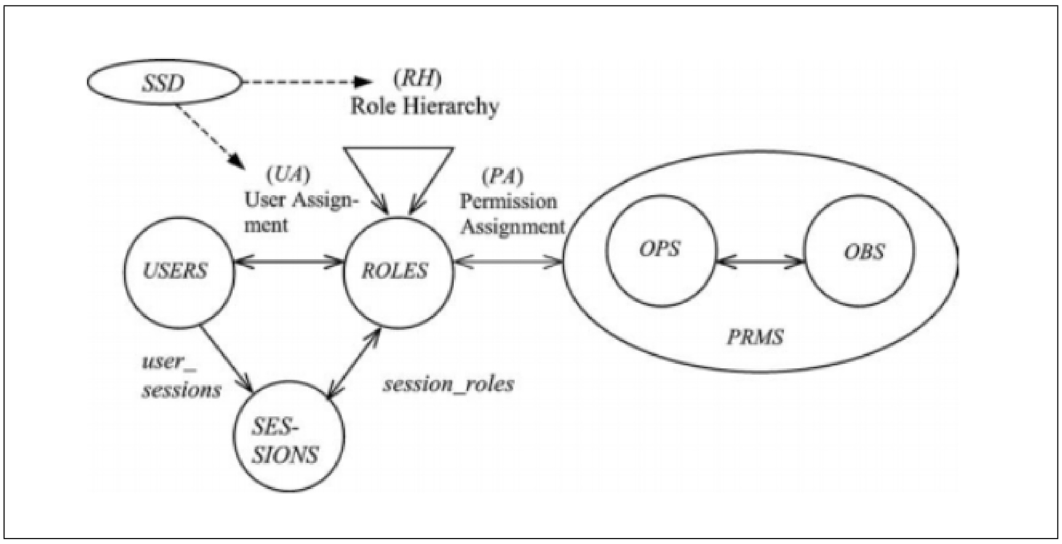


Figure 2.1: Hierarchical Role Based Access Control with Static Separation of Duties as Proposed in [1].

2.1.3.4 Symmetric RBAC

Symmetric RBAC requires permission role review. In this, we can determine the roles which are assigned to particular permissions and we can also determine the permissions which are assigned to a particular role. Permission role review is required in various practical scenarios in policy administration. For example, if a user leaves the organisation or changes his responsibilities within the organisation, then the administrator needs to identify his responsibilities and permissions assigned so that he can revoke user's permissions and roles and can reassign permissions and roles to another user .

2.1.4 Access Control Model Issues and Challenges

The features and limitations associated with the classical access control models are summarised in table 2.1. The current access control models like DAC and MAC do not cover the requirements of most of the organizations for the

Table 2.1: Comparison of Classical Access Control Models

Access Control Model	***Main Features***	***Limitations***
DAC	Based on user identities. Makes use of Access Control Lists (ACL).	No control over multiple copies of the data.
MAC	Based on security levels. These are called clearances for subjects and classification for objects.	Few levels are not sufficient to address the need of many organizations.
RBAC	Based on roles. Need to preassign permission to roles.	Role explosion. No explicit provision to include contextual parameters.

reasons described previously. They are also not very flexible in addressing the needs of different organizations.

RBAC covers most of the issues of DAC and MAC, however it also suffers from the problem of *role explosion*. We need to create a different role in order to fulfil the need of every situation. The problem becomes more challenging for context based access control requirements like at what time request to access a particular resource is initiated, on what day access request is made and from which location the access request is made. In current pervasive computing environment where things changes dynamically, these issues need to be looked at seriously.

Research is going on in this field to address these real life requirements. But most of them proposes a new type of model [8], [9]. Although these models can be implemented but it is difficult to integrate them with existing access control system seamlessly. Most organisation have a RBAC system in place and they seek to improve over their existing underlying model.

Therefore we need an extension in role based access control model so that context information can be included with existing model without doing many changes. Although new ways are being looked at but still there is no standard way to include this information in the RBAC. The expectation to have an automatic access control system is also gaining popularity. This is possible if we can feed the policies and access control model as an input to the system in a formal way. In next section, we cover that all has been done to specify the policy and access control model in a machine understandable format.

2.2 Access Control Policies Specification

In last few year, quite a lot effort has been put to come up with automatic, inference based access control system. This requires to formally specify the authorization policies captured in some access control model framework. The intention is to develop a flexible framework where more information can be added as and when required. XACML [10] is *eXtensible Access Control Markup Language* which is based on XML. It provides a framework which can be used for any organization and not tied up to any specific entity. This is possible due to the useful feature of XML to represent semi-structured data specific to particular organization. It helps in capturing the authorization policy model specification [10]. XACML uses *attributes* to enforce access control. However XACML has a limitation that it is not based on semantics or meanings of the underlying policies. That is why despite being more useful than RBAC, the use of XACML is limited. The Web Ontology Language (OWL) [11] is another way to formally represent the security polices and access control model. The semantic nature of OWL helps in deductive reasoning. That is the reason that researchers have tried combining OWL and XACML [12]. In this work the policy specification is done is separately from that of specification of access control model constructs. For example, managing static and dynamic separation of duties is separated out from the specification of access control polices and their enforcement.

There is another framework proposed which is based on OWL. This framework is KAOS [13]. In this framework, authorization policies are specified using OWL. The framework provide some basic constructs and expect from the user to input the policies using these constructs. In this sense, the user is allowed to feed in the input policy in English but with some constraints. Formal policy representation is done using OWL. Another layer is used to ensure that access is granted as per the polcies. The KAOS multi-layer framework also introduces the concept of guards. A user can use these to get information about policy and then use it for further analysis.

Researchers have come up with some policy languages in which deontic concepts are used. Examples include Rei [14] and Rein [15]. Rei defines various entities like user, action, agent, services *etc.* and uses their properties along with access rights with different credentials. The language is designed to be used where the request can be made from a wide variety of devices. An example scenario is that a university server can be accessed from machines in the labs and from mobile devices in the campus as well. The language defines the concept of policy objects which can be used to capture different rights, dispensations, obligations and prohibitions. It also defines meta policies which are policies about policies. These are used to set the modality and priority among policies whenever there is a conflict among them. Lastly, Rei specifies speech acts to cater for the need of a dynamically changing system. It is used to express and modify policies like delegate, cancel or revoke some access dynamically in the system. Like XACML and OWL, Rei framework is also general purpose and custom implementation can be done without much trouble.

The Rein [15] framework proposes to incorporate N3 [16] rules in the existing Rei framework. The main feature introduced by this framework is to enforce access control based on distributed policies across the web. The process becomes efficient as it is based on the semantic logic inbuilt in OWL. Many organizations use the information specified in other distributed policies to control the uses of their resources. Thus it is useful where we have a distributed network of policies and enforcement is done by considering the data available in all these policies. Rein is based on inference based reasoning and specifically uses Cwm [3].

Although all these policies frameworks are quite useful, however they do not provide a formal synergy between policy language and access control model. This limitation was addressed in ROWLBAC [17] which tries to model RBAC in OWL. They have defined different ontologies to define roles, actions *etc.* and showed how formalism can be incorporated in the automatic access control system. However, they have not addressed the issue of including contextual information in existing RBAC and advocated for a different access control model based on attributes. In our work we improve ROWLBAC further by including context based formalism in the framework.

2.3 Reasoning Based Policy Enforcement

The access control process is guided by high-level policies. The policies are mapped to rules in the framework. When all parts of the rule are true then that particular rule gets fired and its conclusion becomes true. However, in order to check if all parts of a rule are true, we need the process of inference. This is required as all required information may not be directly available in the system and we may need to deduce additional facts from the given data.

The process of policy enforcement should be based on the inference based reasoning. By inference we mean deducing additional facts from the given data. This process helps in capturing the essence of the high level policy. As an example, consider the policy that *PRAN account web page can only be accessed by employees covered under new pension system.* Now if *Suresh* has joined the office on *August 17, 2007* then, based on reasoning, we can deduce that he is covered under *new pension system* and the access would be granted. This is possible because we have this information available in the system that people who join after *January 01, 2004*, are covered under this scheme. As the joining date of *Suresh* is after this date, so we conclude that he is covered by this scheme.

Sometimes information required to deduce addition facts, may be distributed across different data sources. We need to use the semantic nature of the representation and fetch all the required information in order to infer facts about the involved entities. For example, *an organization allows to access its intranet resources by its internal departments or by its partners organizations.* In order to implement this rule, we may make use of IP address of incoming requests and then we use some other data source which keeps the information as which IP address belongs to which organization. This mapping information may be provided by the ISP and may be present at different location. We need to include this information to infer if the request has been generated from an allied parter or the internal department.

Cwm [3] and Euler Yet another proof Engine (EYE) [2] are examples of reasoning engines used extensively. However there is an issue associated with the performance of the reasoner. Based on a study [4] it is proved that EYE is better than Cwm. Both are based on Prolog but the working methodology is different. Both the reasoners are compatible with `N3` and can be used in a distributed way. We experimented with both these and finally decided to go with EYE due to its better performance.

In the following chapter we present an OWL representation of existing RBAC and its proposed extension to incorporate contextual information. We have written OWL ontologies for academic domain. Later in chapter 4 we explain the design and development of a practical access control system and demonstrate how our framework can be useful for organizations.

Chapter 3

Representation of Context Aware RBAC Model in OWL

In this chapter, we first present how the Role Based Access Control can be represented as defined in ROWLBAC [17]. The representation is done in OWL as per the ROWLBAC guidelines. We have defined basic RBAC construct as OWL classes. We have written ontologies in OWL and represented domain ontologies pertaining to academic environment of DTU.

However, the ROWLBAC representation does not contain a way to incorporate contextual information. Therefore we, then, present the extension to the RBAC system for incorporating contextual information. We define constructs related to time, day and location and propose an extension framework for the existing RBAC model by defining contextual ontologies.

3.1 Basic Strategy for RBAC

Basic RBAC constructs include `Subject`, `Object`, `Action` and `Role`. `Action` correspond to the request to access a particular resource or object. Each `Action` is associated with exactly one `Subject` and exactly one `Object`. While `Subject` represents the initiator of an action, `Object` represents the resource which needs to be accessed. Thus, we can say that `Action`s are the access operations which a `Subject` wants to perform on an `Object`. `Role`, on the other hand, is related to the permission and helps in deciding if an `Action` is permitted or not.

We define these basic entities in OWL as follows:

```
Subject  a  owl:Class.
Object   a  owl:Class.
Action   a  owl:Class.
Role     a  owl:Class.
```

Action has been associated with two generic properties: *subject* and *object*. These properties connect the `Action` class to its corresponding `Subject` and `Object`. These are defined as:

```
Action a owl:Class.

subject a owl:Property;
  rdfs:domain Action;
  rdfs:range Subject.

object a rdfs:Property;
  rdfs:domain Action;
  rdfs:range Object.
```

Later, we present a way to add context aware properties with `Action` class. These properties are *accessTime*, *accessLocation* and *accessDay* which represent at what time, from what location and on what day the request is made to perform an action.

`Roles` are associated with `Subject` using two properties: *role* and *activerole*. While the first one states the possible roles which a subject can take, the last tells about which of those roles are actually activated. The access grant decision would be taken based on activated roles. These properties are defined as:

```
Role a owl:Class.

role a owl:Property;
  rdfs:domain Subject;
  rdfs:range Role.

activeRole rdfs:subPropertyOf role.
```

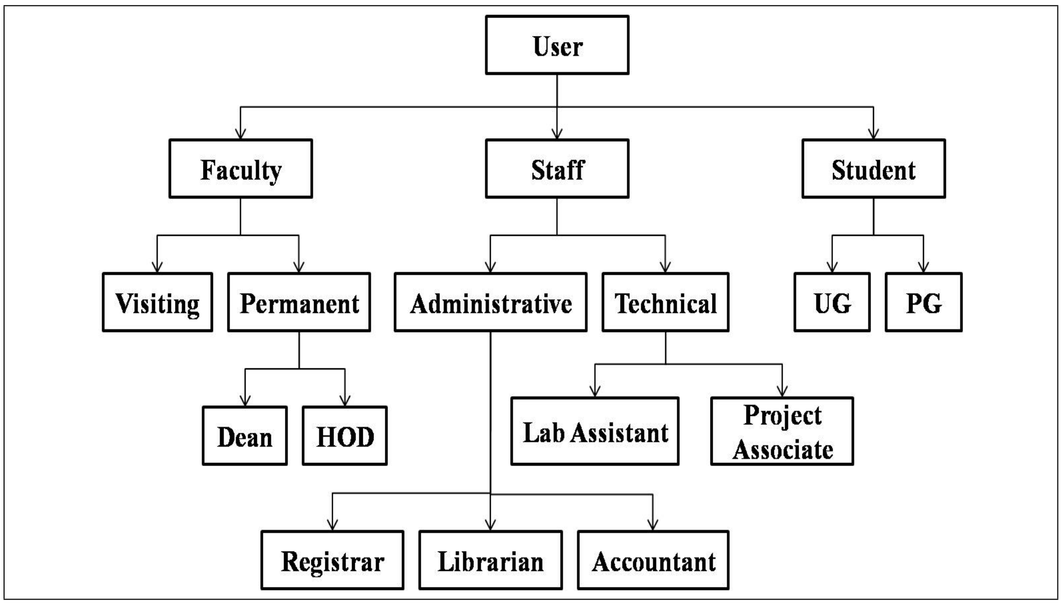


Figure 3.1: Example Role Hierarchy

3.2 Example Domain

We have selected academic environment as an example domain and later show demonstration of the developed RBAC system for the same.

Figure 3.1 shows the role heirarchy for our example domain. In this heirarchy, roles are identified based on the responsibilities and possible activities of different persons in the academic system. The role heirarchy consists of one main class: `User`. `Faculty`, `Student` and `Staff` are subclasses of `User` class. All other classes in the heirarchy are subclasses of these three classes. `Faculty` have two subclasses: `VistingFaculty` and `PermanentFaculty`. `PermanentFaculty` is further divided into two subclasses: `Dean` and `HOD`. Similarly, `Staff` has two subclasses: `AdministrativeStaff` and `TechnicalStaff`. `AdministrativeStaff` is divided in to three subclasses: `Registrar`, `Librarian` and `Accountant`. `TechnicalStaff` has two subclasses: `LabAssistant` and `ProjectAssociate`. `Student` class has two subclasses: `UGstudent` and `PGstudent`.

We have captured only a small portion for the demonstration purpose. More roles can be added in the hierarchy as and when required.

3.3 Defining Role Hierarchies

We have represented role hierarchies using *subRole* property which is defined as:

```
subRole a rdfs:transitiveProperty;
  rdfs:domain Role;
  rdfs:range Role.
```

The *subRole* property connects two roles in a hierarchical way. We have used this property and defined academic domain role hierarchy in following way:

```
User a rbac:Role.

Student a rbac:Role.
Faculty a rbac:Role.
Staff a rbac:Role.

Student rbac:subRole User.

UGStudent rbac:subRole Student.
PGStudent rbac:subRole Student.

Faculty rbac:subRole User.

VisitingFaculty a rbac:Role.
VistingFaculty rbac:subRole Faculty.

PermanentFaculty a rbac:Role.
PermanentFaculty rbac:subRole Faculty.

Dean rbac:subRole PermanentFaculty.
HOD rbac:subRole PermanentFaculty.

Staff rbac:subRole User.
AdministrativeStaff rbac:subRole Staff.
TechnicalStaff rbac:subRole Staff.

LabAssistance rbac:subRole TechnicalStaff.
ProjectAssociate rbac:subRole TechnicalStaff.

Registrar rbac:subRole AdministrativeStaff.
Librarian rbac:subRole AdministrativeStaff.
Accountant rbac:subRole AdministrativeStaff.
```

In this role hierarchy, we have defined `User` as an instance of `Role` class which is a base class and defined in ROWLBAC ontology. All other roles in hierarchy are *subRole*s of the `User`. For example, `Student` is *subRole* of `User`. `Student` is further divided into two roles: `UGStudent` and `PGStudent` which are associated with `Student` using *subRole* property.

This ontology is defined in *appDomain* namespace. In order to access this ontology in other ontologies, the prefix *appDoamin* is used. For example, we can use `Student` role in other domain by writing *appDomain:Student* in the ontology and define the *appDomain* prefix in the beginning of the file.

3.4 Defining Separation of Duties

Separation of duties constraints are used in conflict situations where multiple roles are assigned to single user. Separation of duties constraints are of two types: *Static Separation of Duties* (SSoD) and *Dynamic Separation of Duties* (DSoD). The first constraints tells that one subject cannot have two roles. DSoD constraint tells that although it is possible for the user to have multiple roles but there are some roles which cannot be activated in same session at the same time. SSoD and DSoD are represented using the *ssod* and *dsod* properties in following way:

```
VisitingFaculty rbac:ssod PermanentFaculty.

TA rbac:dsod PGStudent.
```

The first fact asserts that an individual cannot have both `VisitingFaculty` and `PermanentFaculty` roles. The second fact asserts that it is possible for any individual to have both `TA` and `PGStudent` roles but these roles cannot be activated in same session at the same time.

3.5 Defining Permission Assignment

We have used two properties for assigning permissions to roles: *permitted* and *prohibitted*. For example, faculty members and students are allowed to access the result page. This association is represented in following way in our ontology:

```
Faculty rbac:permitted AccessResult.
Student rbac:permitted AccessResult.
```

Here `AccessResult` is defined as an `Action` in our ontology. The *permitted* property connects instances of `Role` class (`Faculty`, `Student`) with instances of `Action` (`AccessResult`) class.

3.6 Representation of Contextual Information

We now propose an extension to the existing ROWLBAC [17] based RBAC framework to include context based information. We need to have this extension as real world situations require to control the flow of information based on context. This includes information like at what time the access request is made, from which location the access request has come from or on what day the request is made to access a specific resource. We now formally present how the context information can be incorporated in existing framework.

3.6.1 Access Control Based on Time

The access control requests based on time are handled by associating *accessTime* property with `Action` class. We introduce this property as follows:

```
accessTime a owl:Property;
  rdfs:domain Action;
  rdfs:range xsd:time.
```

Here, *accessTime* is defined as an OWL property which associates an `Action` with the *time* at which it is requested. We have used datatypes provided by XML schema for this purpose. This datatype is defined in *xsd* namespace.

We also need to use the concept of event which should be characterized by its *start time* and *end time*. The concept of `Event` is defined in *event*

[18] namespace. This namespace also defines other time related classes and properties. We introduce two new properties to defines the *startTime* and *endTime* as follows:

```
startTime
  a owl:DatatypeProperty, owl:FunctionalProperty;
  rdfs:domain event:Event;
  rdfs:range xsd:time.

endTime
  a owl:DatatypeProperty, owl:FunctionalProperty;
  rdfs:domain event:Event;
  rdfs:range xsd:time.
```

Based on above representation, we have defined events like `normalWorkHour`, `morningWorkHour` and `afternoonWorkHour` with different *start* and *end* time in the following way:

```
normalWorkHour a event:Event;
  startTime "09:00:00"^^xsd:time;
  endTime "17:00:00"^^xsd:time.

morningWorkHour a event:Event;
  startTime "09:00:00"^^xsd:time;
  endTime "11:59:00"^^xsd:time.

afternoonWorkHour a event:Event;
  startTime "12:00:00"^^xsd:time;
  endTime "17:00:00"^^xsd:time.
```

In order to check whether the *accessTime* of an `Action` lies within the particular event, we have written specific policy rules which we explain later while discussing the enforcement of the policies.

3.6.2 Access Control Based on Day

Many times organizations put restriction on the use of a resource based on a particular day of week. We introduce another property *accessDay* which tells that on what day of the week the request is made to access a particular resource. We have defined this property as follows:

```
accessDay a owl:Property;
  rdfs:domain Action;
  rdfs:range event:day.
```

The concept of `day` is defined in the event namespace. We later show how the day associated with an `Action` is checked against the allowed day list.

3.6.3 Access Control Based on Group Membership

We have also defined `UserGroup` class in *rbac* namespace which represents the group of user for a specific role. For example, we can group the faculty form CSE department in one `UserGroup` and faculty from EE department in another `UserGroup`.

```
UserGroup a owl:class.
```

For associating a subject with a particular group we have defined the property *hasMember*.

```
hasMember a rdfs:Property;
  rdfs:domain UserGroup;
  rdfs:range Subject.
```

For example, if we want to associate *swamy* with `MtechCSE14` which is a group of *students who joined in 2014 for MTech course in CSE* then we write:

```
MtechCSE14 a rbac:UserGroup;
  rbac:hasMember swamy.
```

3.6.4 Access Control Based on Location

Another important contextual information is the location from which the request has been made. Organizations restrict the access to resources based on the source location. For example, most organizations have some resources which can only be accessed from within the internal network. In some other cases, the information is allowed only for some particular departments within the internal network.

We propose to map the *location* of users based on their IP address. We introduce the concept of `AddressGroup` which groups a set of IP addresses based on the physical locations. This is done with the help of a new property *hasIP*. This property connects the `AddressGroup` to the IP addresses which belong to this group. For example, we define the group for the IP addresses which belong to CSE department as follows:

```
hasIP a rdfs:Property;
  rdfs:domain Action;
  rdfs:range AddressGroup.

AddressCSE a rbac:AddressGroup;
  rbac:hasIP "172.16.124.140".
```

Here we have associated IP address *172.16.124.140* with `AddressCSE` group by using property *hasIP*. We can add many more addresses using this way. We later show how to use the property *hasIP* for checking the membership in a particular `AddressGroup`.

For representing location, *i.e.*, from where the access request is made, we have defined property *accessLocation* whose domain is `Action` class and range is `AddressGroup`. This is the way to incorporate location context in the access request.

```
accessLocation a owl:Property;
  rdfs:domain Action;
  rdfs:range rbac:AddressGroup.
```

3.7 Enforcement of Authorization Policies

In this section, we present how some high level authorization policies, which are expressed in some natural language, can be converted in OWL rules and enforced. For writing policy rules, `N3` notation has been used due to its better readability. These rules are enforce using the EYE reasoner which is based on inference based reasoning. We now present some example policies and their enforcement.

1. As the first example consider a higher level policy statement says that *access to result page is allowed by students and faculty from 10:00 AM to 5:00PM.* For enforcing this policy, we write the corresponding rule as:

```
{   ?ACT a  ?REQACTION;
       subject ?SUB;
       accessTime ?T.
    ?REQACTION a Action.
    ?SROLE permitted ?REQACTION.
    ?SUB activeRole ?SROLE.
    normalWorkHour time:includes ?T.
}  => {?ACT a PermittedAction;
                    role ?SROLE;
                    action?REQaction;
                    Subject ?SUB.
         }
```

   In specifying policy rules if all the statements on the left hand side of the implication operator are true then the rule will be fired and we assert (consider true) the facts specified in the right hand side of the implication operator.

   The explanation of the above rule is as follows. We say that `Act` is an `Action`. This is specified in two steps. We first say that `Act` is `REQACTION` and then specify that `REQACTION` is a type of `Action`. The associated subject of the `Action` is `SUB` and associated access time is `T`. We now check if there is any role (`SROLE`) which is permitted to perform the requested action and it is one of the activated roles of the subject. We also check if the access time `T` lies within the normal work hour range. If all these conditions are satisfied then the requested action will be a permitted action.

   Above rule includes the context parameter access time which tells the time at which the request is made by a subject to access the specific object. In above rule, we need to check if access time falls within `normalWorkHour`. We have to compare access time with respect to the *start time* and *end time* of `normalWorkHour`. We have written rules to check this thing as:

```
{
   ?e startTime ?begin.
   ?A a ?REQACTION;
      accessTime ?T.
   ?REQACTION a Action.
   ?begin math:equalTo ?T .
} =>
{
    ?e time:includes ?T .
} .
```

Above rule checks the access time with respect to *start time* of the event. If they happen to be equal then the access time is within the time limits of the event. Here *equalTo* property is defined in *math* namespace and *includes* property is defined in *time* namespace.

Following rule checks the access time against *end time* of the event.

```
{
    ?e endTime ?ending.
    ?A a ?REQACTION;
       accessTime ?T.
    ?REQACTION a Action.
    ?ending math:equalTo ?T .
} =>
{
    ?e time:includes ?T.
} .
```

We need to write one more rule to check if the access time is between the *start time* or *end time* of the event. This is done as follows:

```
{
    ?e startTime ?begin.
    ?e endTime ?ending.
    ?A a ?REQACTION;
       accessTime ?T.
    ?REQACTION a Action.
    ?begin math:lessThan ?T .
    ?ending math:greaterThan ?T .
} =>
{
    ?e time:includes ?T.
} .
```

2. As another example, we now present how to write policy rules which restrict the access based on the location. For example, consider the policy rule *access to a web page is allowed if the request has come from the CSE department.* This policy rule is written in following way:

```
{
  ?ACT a ?REQACTION;
     subject ?SUB;
     object  ?OBJ;
     hasIP ?IP.
 ?REQACTION a Action.
 ?SROLE permitted ?REQACTION.
 ?SUB activeRole ?SROLE.
 data:AddressCSE hasIP ?IP.
} => {?ACT a PermittedAction;
             role ?SROLE;
             action ?REQACTION;
             subject ?SUB.
}.
```

   In above rule, the requested action has associated subject `SUB` and has the IP address `IP`. Apart from checking the required activated role as explained in previous example, we also check if the source IP address belongs to the CSE department machines. The *hasIP* property associates requested action with the IP address of machine from where request is generated. For enforcing policies which require access from particular IP we define a set of IP addresses which belong to some particular location. In above example this set is `AddressCSE` which is defined in *data* namespace.

3. Organizations also need a way to enforce access control based on days of a week. Consider a high level policy rule saying that *access to faculty page is allowed on Thursday and Friday by faculty of Electrical Engineering Department.* For enforcing these types of policies, it requires to check whether access to an object by a subject from a particular department is allowed on particular day. We have written this rule in following manner:

```
{
  ?ACT a ?REQACTION;
```

```
     subject ?SUB;
     object  ?OBJ;
     accessDay ?d.
 ?REQACTION a Action.
 ?SROLE permitted ?REQACTION.
 ?SUB activeRole ?SROLE.
 data:FacultyEE hasMember ?SUB.
 ?d list:in ("Thursday" "Friday").
} => {?ACT a PermittedAction;
            role ?SROLE;
            action ?REQACTION;
            subject ?SUB.
        }.
```

In above rule, the requested action has associated subject `SUB` and context day `d`. In above example, we are associating context information with the subject in the form of group membership. In this, we check if a user from a particular `UserGroup` can access the requested page on a particular day. Here we are checking whether the subject is a member of `FacultyEE UserGroup`. Similarly for the policy statement *access to faculty page is allowed on Monday, Tuesday and Wednesday by the faculty of Computer Science and Engineering Department*, the rule is written in following way:

```
{
  ?ACT a ?REQACTION;
     subject ?SUB;
     object  ?OBJ;
     accessDay ?d.
 ?REQACTION a Action.
 ?SROLE permitted ?REQACTION.
 ?SUB activeRole ?SROLE.
 data:FacultyCSE hasMember ?SUB.
 ?d list:in ("Monday" "Tuesday" "Wednesday").
} => {?ACT a PermittedAction;
           role ?SROLE;
           action ?REQACTION;
           subject ?SUB.
        }.
```

4. In organizations where multiple roles are assigned to single person there need to be some constraints on activation of roles. These constraints are driven by separation of duties as explained earlier. We have written rule for checking *DSoD* constraint as follows:

```
{
  ?ACT a ?RACTIVATEROLE;
     subject ?SUB;
     object ?NEWROLE.
  ?RACTIVATEROLE a Action.
  ?SUB activeRole ?R.
  ?R dsod ?NEWROLE.
} => { ?ACT a prohibittedAction;
            subject ?SUB;
            object  ?NEWROLE.
        }.
```

In above rule, the requested action has associated subject `SUB` and the proposed new role `NEWROLE`. We check if there is any other existing activated role which has a dynamic separation of duties constraint with `NEWROLE`. In that case the requested would be prohibited.

In this chapter, we have formally presented the extension to the RBAC framework to include contextual information. We have presented sufficient examples to show how context based policies can be enforced. In next chapter, we describe the developed access control system for DTU examination portal. We also demonstrate how our framework can be integrated into practical systems.

Chapter 4

Access Control System for Academic Environment

In this chapter, we present how the theoretical framework introduced in previous chapter can be used to implement a practical access control system of interest. We have implemented an access control system for academic environment. In particular, the examination portal of DTU is used for demonstration. The developed system is context aware and fulfils most of the realtime access control requirements. In order to enforce policy based access control user roles and contextual information are used primarily. These roles are assigned to users based on the activities in which he or she is involved as part of Delhi Technological University. Context information associated with access request can be *time*, *day* of week or *location* from where request is initiated. This system clearly demonstrate the potential of our representation framework for access control in practical scenarios.

4.1 System design

The overall design of the developed system is shown in figure 4.1. Implemented access control system shows that how it can be used in practical situations. When some user wants to access some resource (a web page in our system), she would make a request for that resource using her client (*e.g.* browser). The incoming requests are handled by the front end at the server. The front end authenticates the user and, according to the request made, it creates a session. The session contains information about what is requested along with other access control parameters. The front end of the server extracts relevant information from the incoming request. The relevant information, for example, can be parameters like subject, object, time, day or location. The front end also communicates with policy database to get authorization policies of the organization and invoke the reasoner. The reasoner also takes the access control model specification and application related information (*roles*, *permissions etc.*) as input. Reasoner does the necessary work required for inference based reasoning and produces the output which contains the authorization decision. Based on the output of the reasoner, the desired information is then returned to the client.

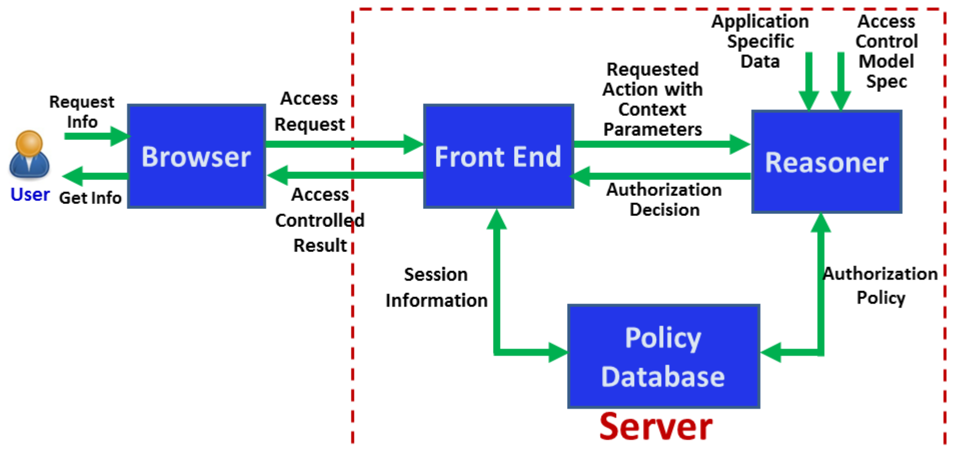


Figure 4.1: The Overall Design of the Developed Access Control System

For example, consider the case when a user *swamy* wants to access the *result page* at the examination portal. As a first step the user would login to the system using his credentials. These credentials are then authenticated by the server. If the authentication is successful then the server creates a session for the user *swamy*. It also communicates with the policy database and records the currently activated role of the user.

When *swamy* clicks on the result page link then the front end of the server creates the action which contains the subject (*swamy*), object (*result page*) and context information (*time*, *day* and *location*) as parameters and add this request to currently activated session for the user *swamy*. Server then invokes the reasoner. Reasoner takes input as access control model specification, application specific data, authorization policies and session information. The reasoner performs the inference based reasoning and produces the output which contains the authorization decision. This output is then send back to the front end of the server. The front end analyses the output and, based on the authorization decision, it returns the result to the browser of the user. Thus if authorization decision says that requested action is permitted then server renders the desired page to the browser of the user and if requested action is not permitted, according to authorization decision, then server sends the message to the browser that access to the requested page is denied along with possible reasons.

4.2 Access Control System for Examination Portal at DTU

We have selected the examination portal of DTU for demonstration of the access control system. As of now the examination portal at Delhi Technological University does not have any access control system in place. We have added few more pages to this portal for demonstrating our system. In the following, we explain the functionality of these pages along with description of the developed access control system.

4.2.1 Login Page

We have added *login page* on top of the examination portal of DTU for authentication of user. We require this page because, for access control, there is a need to identify the user. From user credentials, we get the information that who has logged in to the system, who wants to access the system resources and what permissions he has. The page is shown in figure 4.2.

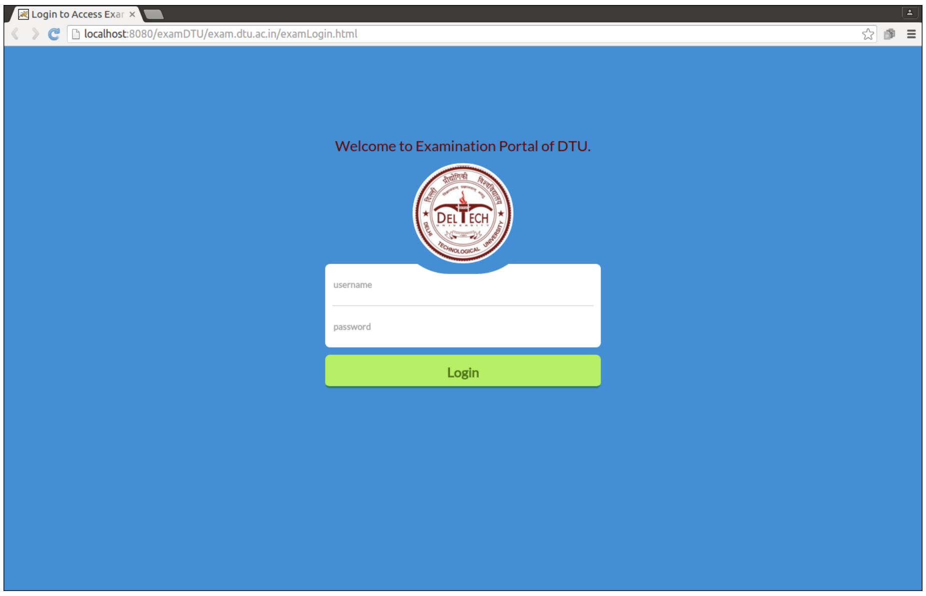

Login to Access Exar
localhost:8080/examDTU/exam.dtu.ac.in/examLogin.html
Welcome to Examination Portal of DTU.
username
password
Login

Figure 4.2: Login Page to Access Examination Portal of DTU

Login page helps to get the information which are necessary to decide access control. For example, if a user has an administrator role activated then he is allowed to access all system resources. However, if the user is not an administrator then his access to the system resources would be based on his possible and activated roles and associated context information.

4.2.2 Administrator Page

If the *logged-in* user is an administrator then, after successful login, administrator page will be displayed as shown if figure 4.3. Administrator can go directlly to the examination portal by clicking on *Go To Examination Portal* link and can access any page of the examination portal he wants to access. Additionally, administrator can also manage the authorization policies.

Every organisation requires management of policies for effective access control. Policy management includes viewing the existing policies, modifying the existing policies and writing new policies. We have created policy management page for this purpose. Figure 4.4 shows the snapshot of *policy management page*.

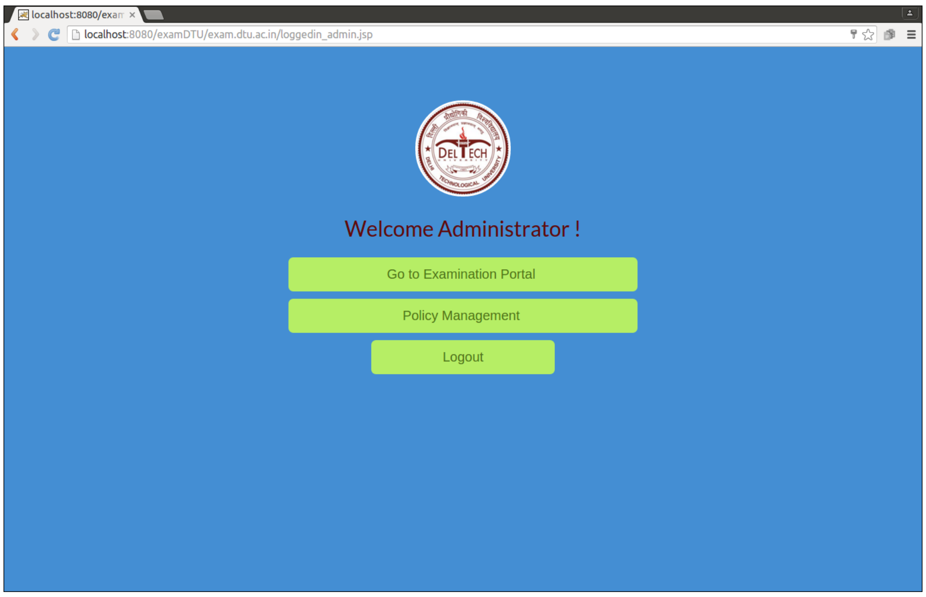


Figure 4.3: After Login Page for Administrator to Access Examination Portal of DTU

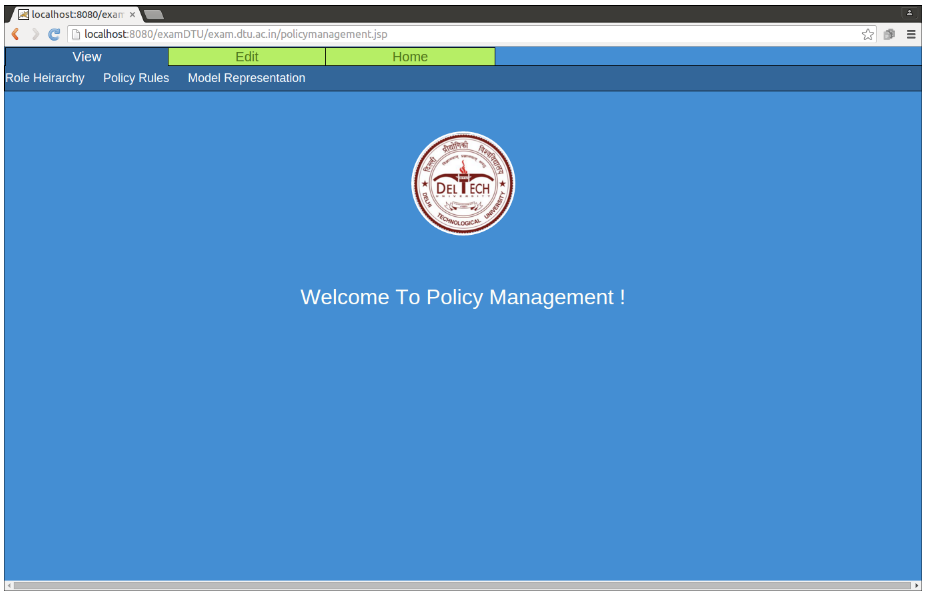
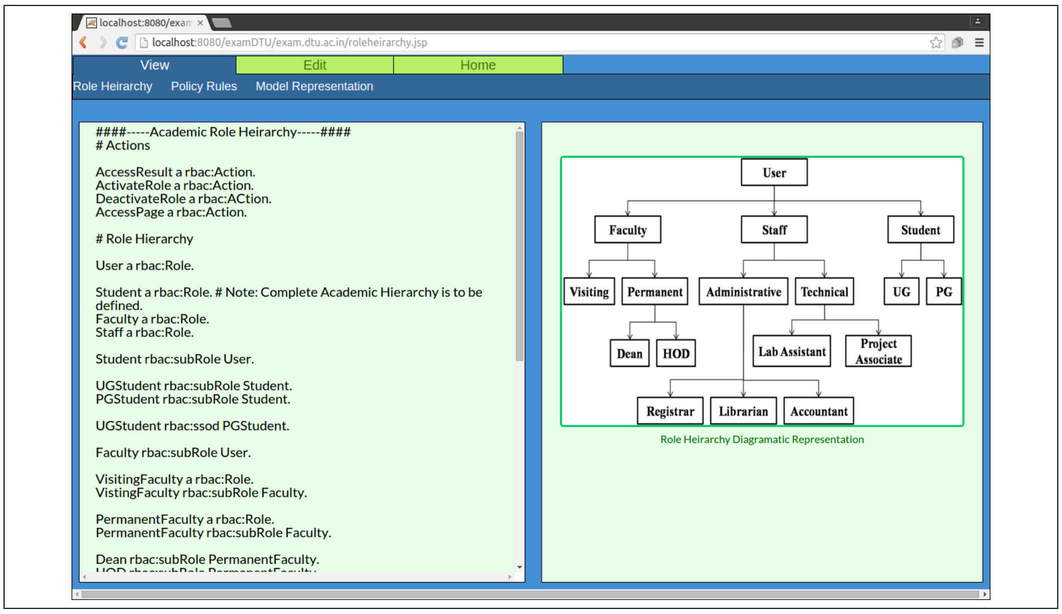

localhost:8080/exam
localhost:8080/examDTU/exam.dtu.ac.in/policymanagement.jsp
View
Edit
Home
Role Heirarchy
Policy Rules
Model Representation
Welcome To Policy Management !

Figure 4.4: Policy Management Main Page

We have provided *view menu* and *edit menu* on this page. Using *view* *menu* existing policies can be viewed and analysed by the administrator while *edit menu* is designed so that the administrator can edit existing policies while. For going back to administrator page, a *home link* is provided.

In the *view menu*, we have three sub menus for viewing role heirarchy, existing *rbac* policies and *rbac* model representation. Figure 4.5 shows the policy management page with *view menu* selected for role heirarchy.



```
####-----Academic Role Heirarchy-----####
# Actions

AccessResult a rbac:Action.
ActivateRole a rbac:Action.
DeactivateRole a rbac:ACtion.
AccessPage a rbac:Action.

# Role Hierarchy

User a rbac:Role.

Student a rbac:Role. # Note: Complete Academic Hierarchy is to be
defined.
Faculty a rbac:Role.
Staff a rbac:Role.

Student rbac:subRole User.

UGStudent rbac:subRole Student.
PGStudent rbac:subRole Student.

UGStudent rbac:ssod PGStudent.

Faculty rbac:subRole User.

VisitingFaculty a rbac:Role.
VistingFaculty rbac:subRole Faculty.

PermanentFaculty a rbac:Role.
PermanentFaculty rbac:subRole Faculty.

Dean rbac:subRole PermanentFaculty.
```

Figure 4.5: Policy Management Page for View Role Hierarchy

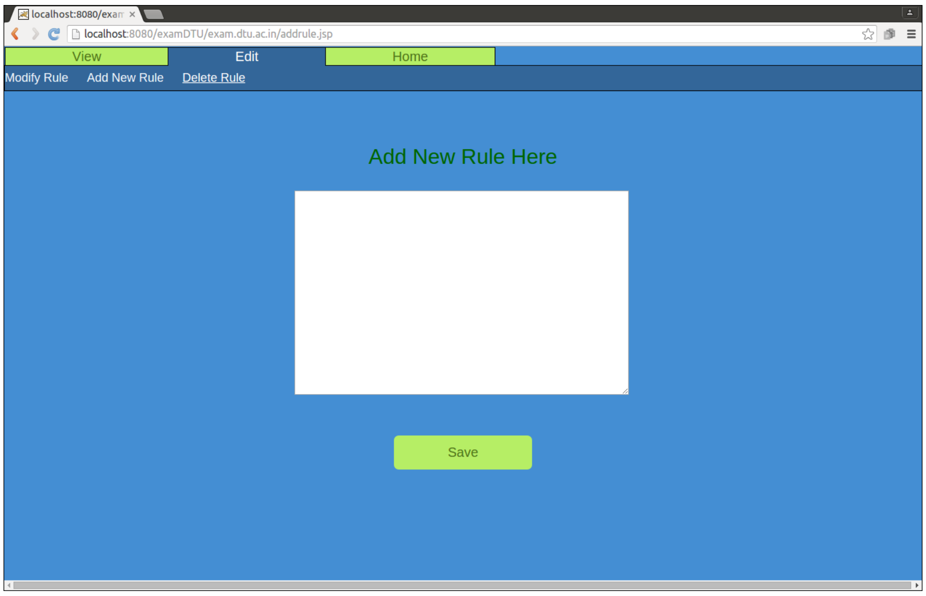
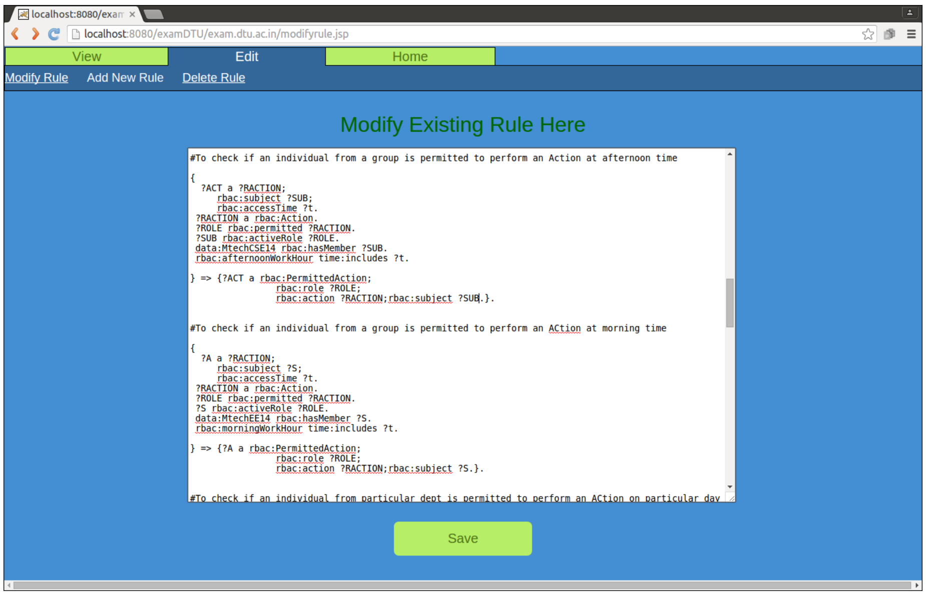


Figure 4.6: Policy Management Page for Add New Rule

We have provided *edit menu* on *policy management page* so that administrator can edit existing policies. Administrator can add new rules or modify existing rules using the sub menus provided in the *edit menu*. Figure 4.6 and figure 4.7 show these functionalities.


localhost:8080/exam
localhost:8080/examDTU/exam.dtu.ac.in/modifyrule.jsp
View
Edit
Home
Modify Rule
Add New Rule
Delete Rule
Modify Existing Rule Here
Save

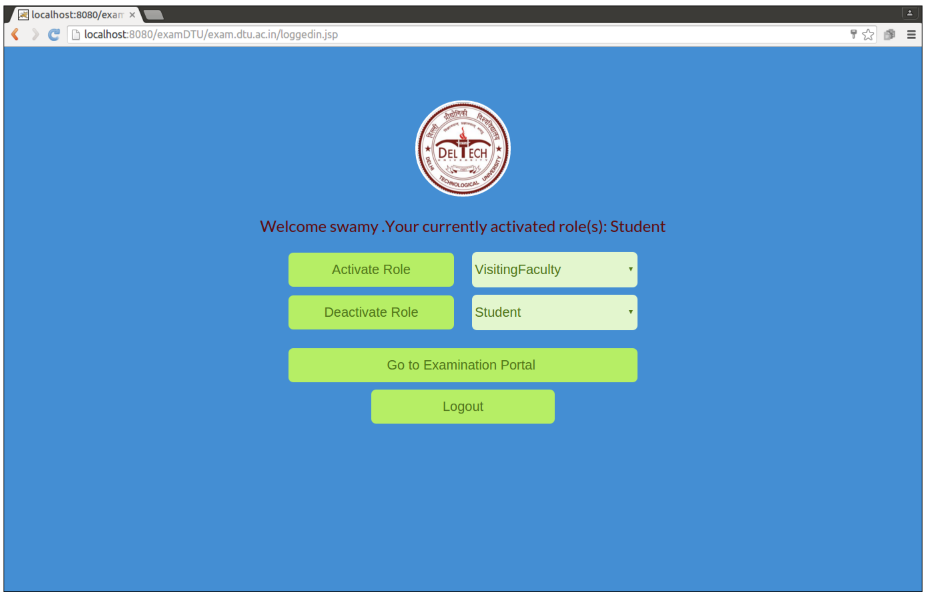
```
#To check if an individual from a group is permitted to perform an Action at afternoon time

{
  ?ACT a ?RACTION;
     rbac:subject ?SUB;
     rbac:accessTime ?t.
 ?RACTION a rbac:Action.
 ?ROLE rbac:permitted ?RACTION.
 ?SUB rbac:activeRole ?ROLE.
 data:MtechCSE14 rbac:hasMember ?SUB.
 rbac:afternoonWorkHour time:includes ?t.

} => {?ACT a rbac:PermittedAction;
                rbac:role ?ROLE;
                rbac:action ?RACTION;rbac:subject ?SUB.}.


#To check if an individual from a group is permitted to perform an ACtion at morning time

{
  ?A a ?RACTION;
     rbac:subject ?S;
     rbac:accessTime ?t.
 ?RACTION a rbac:Action.
 ?ROLE rbac:permitted ?RACTION.
 ?S rbac:activeRole ?ROLE.
 data:MtechEE14 rbac:hasMember ?S.
 rbac:morningWorkHour time:includes ?t.

} => {?A a rbac:PermittedAction;
                rbac:role ?ROLE;
                rbac:action ?RACTION;rbac:subject ?S.}.


#To check if an individual from particular dept is permitted to perform an ACtion on particular day
```

Figure 4.7: Policy Management Page for Modify Existing Rule

4.2.3 Activation and Deactivation of Roles

A user should be able to activate or deactivate her roles during her session. However, this process is subject to the constraint imposed by Dynamic Separation of Duties. We explain that part in the next section.

We have provided options for activation and deactivation of roles for users. This is applicable for users other then the administrator. If the user is not an administrator then after successful login, the page shown in figure 4.8 is displayed. This page displays the default activated role of user and provides the option to activate role from his possible roles. However, not all combinations of multiple roles can be simultaneously activated due to dynamic separation of duties constraint. We have also provided option for user to deactivate his or her active role. Each activated role is associated with a set of permissions and helps in deciding access grant along with contextual information to access different pages of the examination portal. For going to examination portal we have provided separate button named *Go to examination portal*.



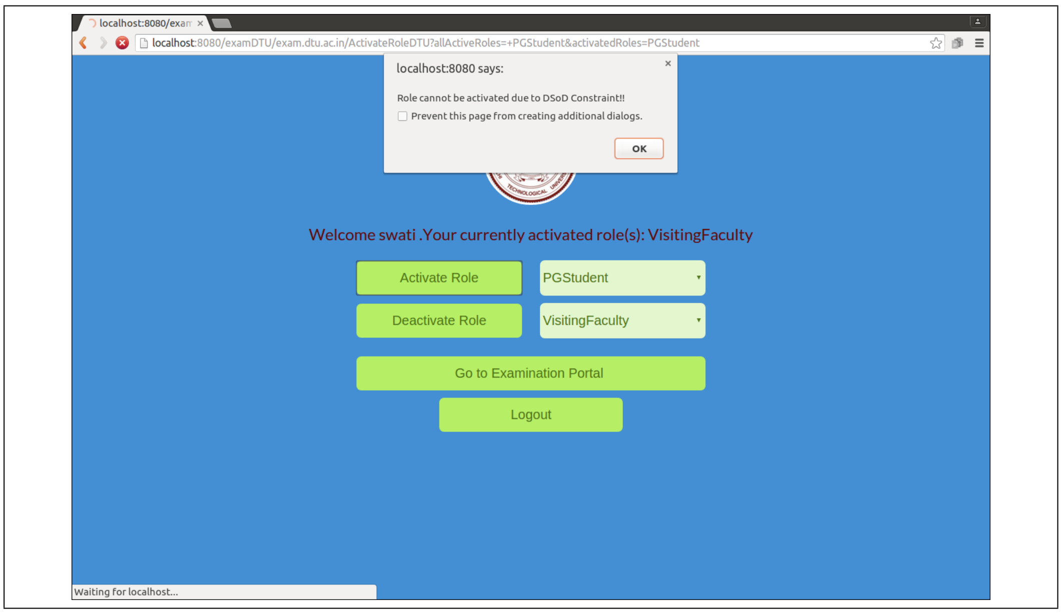
Welcome swamy .Your currently activated role(s): Student

Figure 4.8: After Login Page to Access Examination Portal of DTU

4.2.4 Dynamic Separation of Duties

As explained in previous section, we check for dynamic separation of duties constraint during activation of roles. In order to implement this check, we have written separate rules which are already discussed in chapter 3.

For example, consider the case where *swati* is a subject and whose possible roles are `VisitingFaculty` and `PGStudent`. There exist dynamic separation of duty constraint between these two roles as these two roles can not be simultaneously activated. In our implementation if *swati* has activated one role (for example, `VisitingFaculty` here) and want to activate another role (`PGStudent`) at the same time then this action is not allowed. The figure 4.9 shows that the same page will be displayed with a warning message.


localhost:8080/examDTU/exam.dtu.ac.in/ActivateRoleDTU?allActiveRoles=+PGStudent&activatedRoles=PGStudent

localhost:8080 says:

Role cannot be activated due to DSoD Constraint!!

☐ Prevent this page from creating additional dialogs.

Welcome swati .Your currently activated role(s): VisitingFaculty

Waiting for localhost...

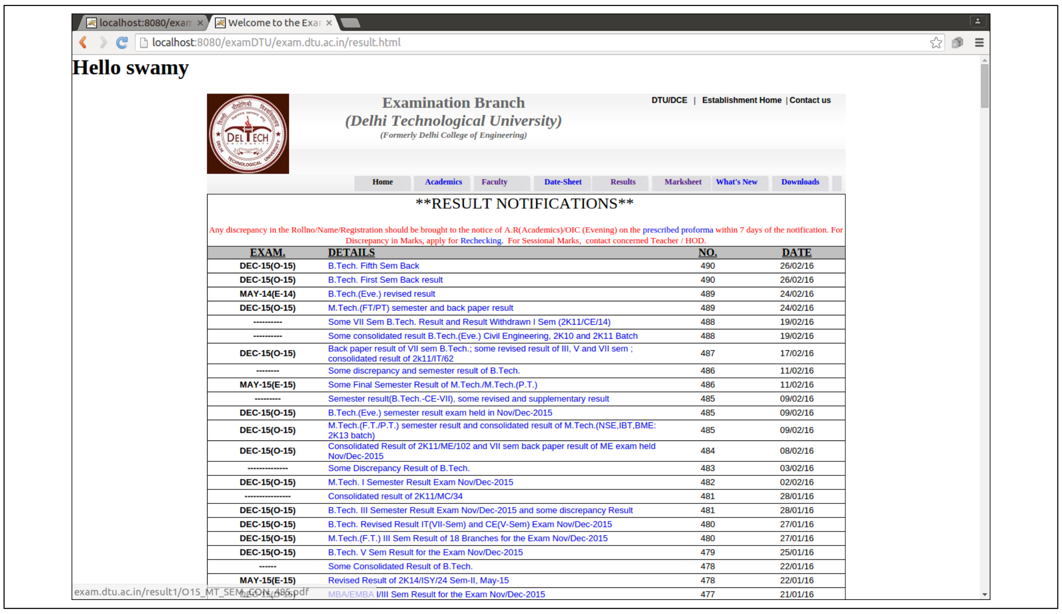
Figure 4.9: Page Showing DSOD Constraint

4.2.5 Access Control Based on Time

Many times, in practical scenarios, the access permission is be based on the time at which the access request is presented to the system. This may be because the organization may not have enough capability to serve all users at the same time.

We have enforced *time* based access control on *result page* of examination portal. If a user, other than the administrator, clicks on the *result page* link, then the server creates the action which contains the information of *subject*, *object* and *access time* at which request is presented to the server. Server also identifies the user group from which user belongs to, and sends this information as an input to the reasoner. Reasoner then performs the inferencing and based on the role of the user and the time at which request is presented, produces the authorization decision. If the decision says that requested action is permitted then server renders the desired page.

For example our policy says that the access to *result page* is allowed to students from `MtechCSE14` group between `9:00 AM` to `11:59AM` and for students from `MtechEE14` group between `12:00 PM` to `5:00 PM`. Thus if a student from `MtechCSE` group wants to access the *result page* at `11:00 AM` he would see the page displayed as shown on figure 4.10.



Hello swamy

Examination Branch
(Delhi Technological University)
(Formerly Delhi College of Engineering)

RESULT NOTIFICATIONS

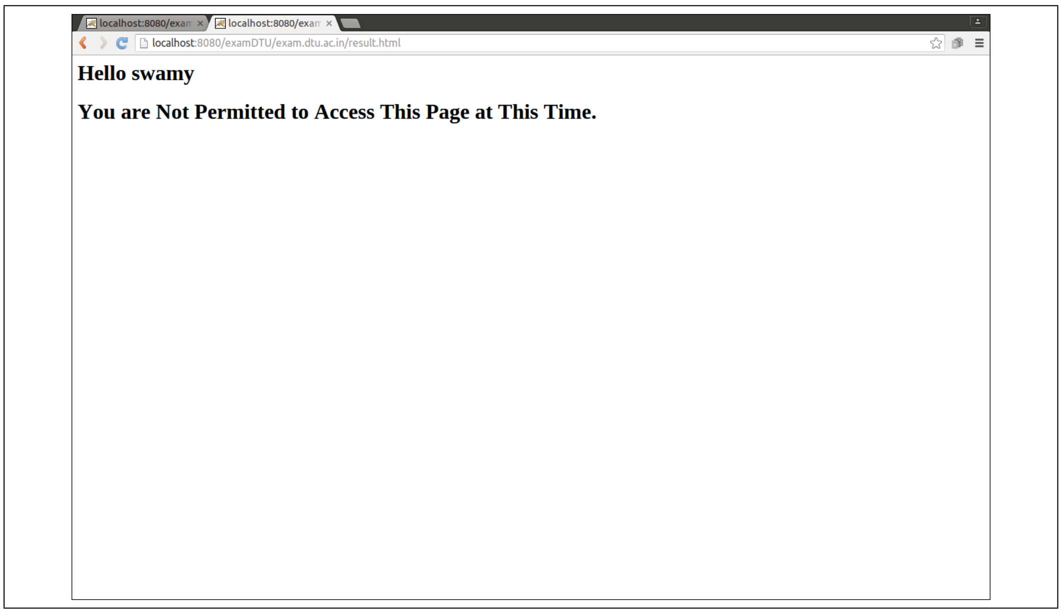
Any discrepancy in the Rollno/Name/Registration should be brought to the notice of A.R(Academics)/OIC (Evening) on the prescribed proforma within 7 days of the notification. For Discrepancy in Marks, apply for Rechecking. For Sessional Marks, contact concerned Teacher / HOD.

EXAM.	DETAILS	NO.	DATE
DEC-15(O-15)	B.Tech. Fifth Sem Back	490	26/02/16
DEC-15(O-15)	B.Tech. First Sem Back result	490	26/02/16
MAY-14(E-14)	B.Tech.(Eve.) revised result	489	24/02/16
DEC-15(O-15)	M.Tech.(FT/PT) semester and back paper result	489	24/02/16
----------	Some VII Sem B.Tech. Result and Result Withdrawn I Sem (2K11/CE/14)	488	19/02/16
----------	Some consolidated result B.Tech.(Eve.) Civil Engineering, 2K10 and 2K11 Batch	488	19/02/16
DEC-15(O-15)	Back paper result of VII sem B.Tech.; some revised result of III, V and VII sem ; consolidated result of 2k11/IT/62	487	17/02/16
--------	Some discrepancy and semester result of B.Tech.	486	11/02/16
MAY-15(E-15)	Some Final Semester Result of M.Tech./M.Tech.(P.T.)	486	11/02/16
---------	Semester result(B.Tech.-CE-VII), some revised and supplementary result	485	09/02/16
DEC-15(O-15)	B.Tech.(Eve.) semester result exam held in Nov/Dec-2015	485	09/02/16
DEC-15(O-15)	M.Tech.(F.T./P.T.) semester result and consolidated result of M.Tech.(NSE,IBT,BME: 2K13 batch)	485	09/02/16
DEC-15(O-15)	Consolidated Result of 2K11/ME/102 and VII sem back paper result of ME exam held Nov/Dec-2015	484	08/02/16
--------------	Some Discrepancy Result of B.Tech.	483	03/02/16
DEC-15(O-15)	M.Tech. I Semester Result Exam Nov/Dec-2015	482	02/02/16
----------------	Consolidated result of 2K11/MC/34	481	28/01/16
DEC-15(O-15)	B.Tech. III Semester Result Exam Nov/Dec-2015 and some discrepancy Result	481	28/01/16
DEC-15(O-15)	B.Tech. Revised Result IT(VII-Sem) and CE(V-Sem) Exam Nov/Dec-2015	480	27/01/16
DEC-15(O-15)	M.Tech.(F.T.) III Sem Result of 18 Branches for the Exam Nov/Dec-2015	480	27/01/16
DEC-15(O-15)	B.Tech. V Sem Result for the Exam Nov/Dec-2015	479	25/01/16
------	Some Consolidated Result of B.Tech.	478	22/01/16
MAY-15(E-15)	Revised Result of 2K14/ISY/24 Sem-II, May-15	478	22/01/16
DEC-15(O-15)	MBA/EMBA I/III Sem Result for the Exam Nov/Dec-2015	477	21/01/16

exam.dtu.ac.in/result1/O15_MT_SEM_CON_485.pdf

Figure 4.10: Result Page Displayed at Particular Time

Similarly when a student from the `MtechEE` group wants to access the result page at `11:00 AM`, he will get the access denied page as shown in figure 4.11.


localhost:8080/examDTU/exam.dtu.ac.in/result.html

Hello swamy

You are Not Permitted to Access This Page at This Time.

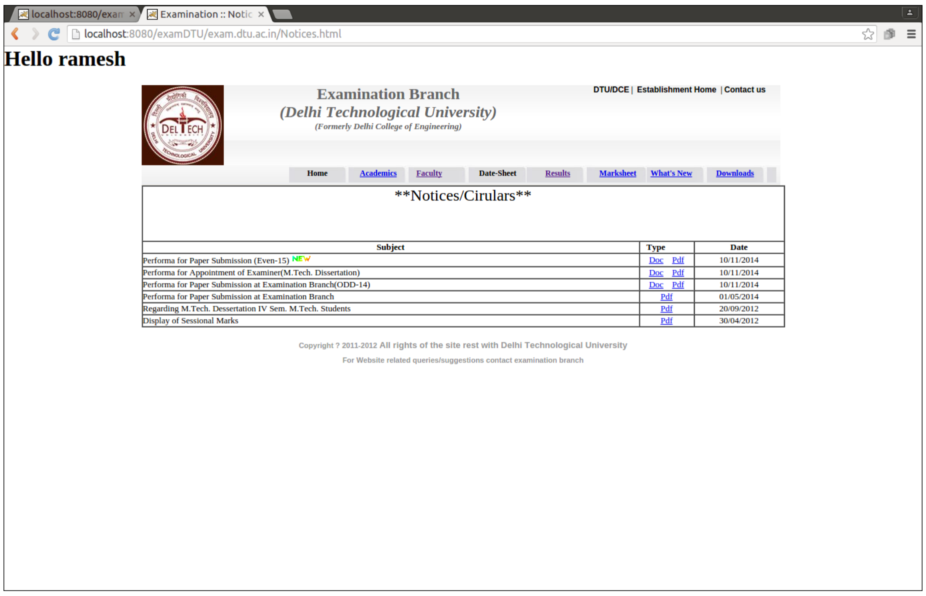
Figure 4.11: Access denied to Result Page

4.2.6 Access Control Based on days of Week

Access restriction based on a particular `day` of a week is another way to distribute the work load of the system. We have incorporated access control on the basis of on what day of week access request is generated by a particular subject to access an object. We have implemented the access control based on day on the Faculty page of the examination portal.

For example, out policy rule says that `Faculty` from CSE department can access the *faculty page* on `Monday`, `Tuesday` and `Wednesday` while `Faculty` from EE department can access the *faculty page* on `Thursday` and `Friday`.

Figure 4.12 shows that the *faculty page* is displayed when accessed by a faculty of CSE department on `Monday`. On the other hand, access to the *faculty page* is denied when accessed by a faculty of EE department on `Monday` as shown in figure 4.13.



localhost:8080/examDTU/exam.dtu.ac.in/Notices.html

Hello ramesh

Examination Branch
(Delhi Technological University)
(Formerly Delhi College of Engineering)

DTU/DCE | Establishment Home | Contact us

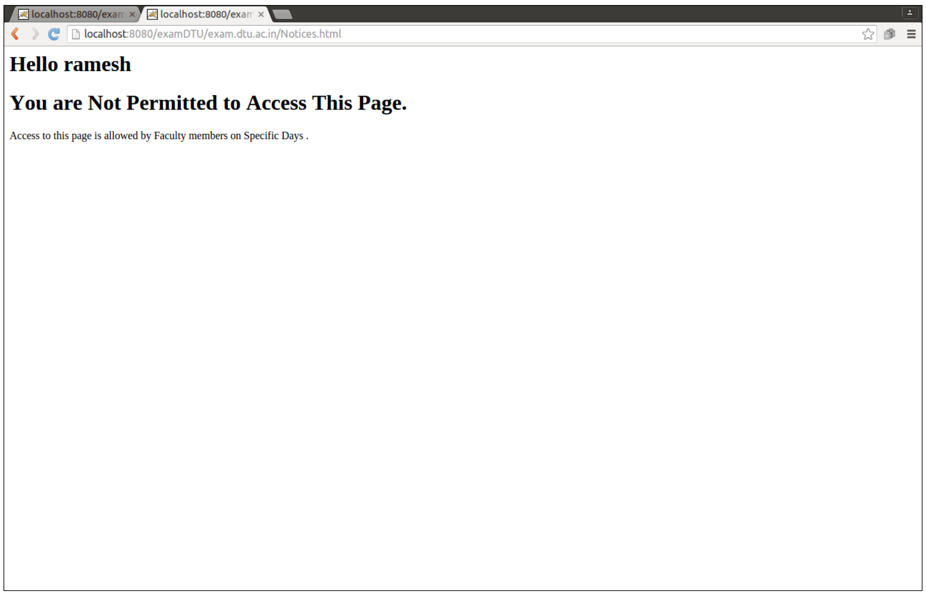
Home | Academics | Faculty | Date-Sheet | Results | Marksheet | What's New | Downloads

Notices/Cirulars

Subject	Type	Date
Performa for Paper Submission (Even-15) NEW	Doc Pdf	10/11/2014
Performa for Appointment of Examiner(M.Tech. Dissertation)	Doc Pdf	10/11/2014
Performa for Paper Submission at Examination Branch(ODD-14)	Doc Pdf	10/11/2014
Performa for Paper Submission at Examination Branch	Pdf	01/05/2014
Regarding M.Tech. Dessertation IV Sem. M.Tech. Students	Pdf	20/09/2012
Display of Sessional Marks	Pdf	30/04/2012

Copyright ? 2011-2012 All rights of the site rest with Delhi Technological University
For Website related queries/suggestions contact examination branch

Figure 4.12: Faculty Page When Accessed on Particular day



localhost:8080/examDTU/exam.dtu.ac.in/Notices.html

Hello ramesh

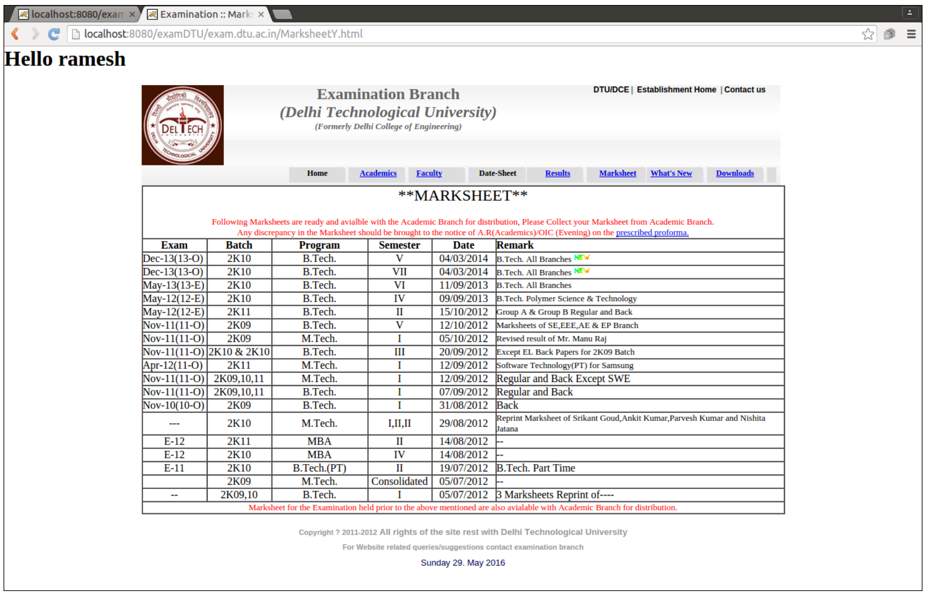
You are Not Permitted to Access This Page.

Access to this page is allowed by Faculty members on Specific Days .

Figure 4.13: Access Denied to Faculty Page

4.2.7 Access Control Based on Location

We have implemented location based access control over the *marksheet page* of examination portal at DTU.



Hello ramesh

Examination Branch
(Delhi Technological University)
(Formerly Delhi College of Engineering)

MARKSHEET

Following Marksheets are ready and avialble with the Academic Branch for distribution, Please Collect your Marksheet from Academic Branch.
Any discrepancy in the Marksheet should be brought to the notice of A.R(Academics)/OIC (Evening) on the prescribed proforma.

Exam	Batch	Program	Semester	Date	Remark
Dec-13(13-O)	2K10	B.Tech.	V	04/03/2014	B.Tech. All Branches NEW
Dec-13(13-O)	2K10	B.Tech.	VII	04/03/2014	B.Tech. All Branches NEW
May-13(13-E)	2K10	B.Tech.	VI	11/09/2013	B.Tech. All Branches
May-12(12-E)	2K10	B.Tech.	IV	09/09/2013	B.Tech. Polymer Science & Technology
May-12(12-E)	2K11	B.Tech.	II	15/10/2012	Group A & Group B Regular and Back
Nov-11(11-O)	2K09	B.Tech.	V	12/10/2012	Marksheets of SE,EEE,AE & EP Branch
Nov-11(11-O)	2K09	M.Tech.	I	05/10/2012	Revised result of Mr. Manu Raj
Nov-11(11-O)	2K10 & 2K10	B.Tech.	III	20/09/2012	Except EL Back Papers for 2K09 Batch
Apr-12(11-O)	2K11	M.Tech.	I	12/09/2012	Software Technology(PT) for Samsung
Nov-11(11-O)	2K09,10,11	M.Tech.	I	12/09/2012	Regular and Back Except SWE
Nov-11(11-O)	2K09,10,11	B.Tech.	I	07/09/2012	Regular and Back
Nov-10(10-O)	2K09	B.Tech.	I	31/08/2012	Back
---	2K10	M.Tech.	I,II,II	29/08/2012	Reprint Marksheet of Srikant Goud,Ankit Kumar,Parvesh Kumar and Nishita Jatana
E-12	2K11	MBA	II	14/08/2012	--
E-12	2K10	MBA	IV	14/08/2012	--
E-11	2K10	B.Tech.(PT)	II	19/07/2012	B.Tech. Part Time
	2K09	M.Tech.	Consolidated	05/07/2012	--
--	2K09,10	B.Tech.	I	05/07/2012	3 Marksheets Reprint of----

Marksheet for the Examination held prior to the above mentioned are also avialable with Academic Branch for distribution.

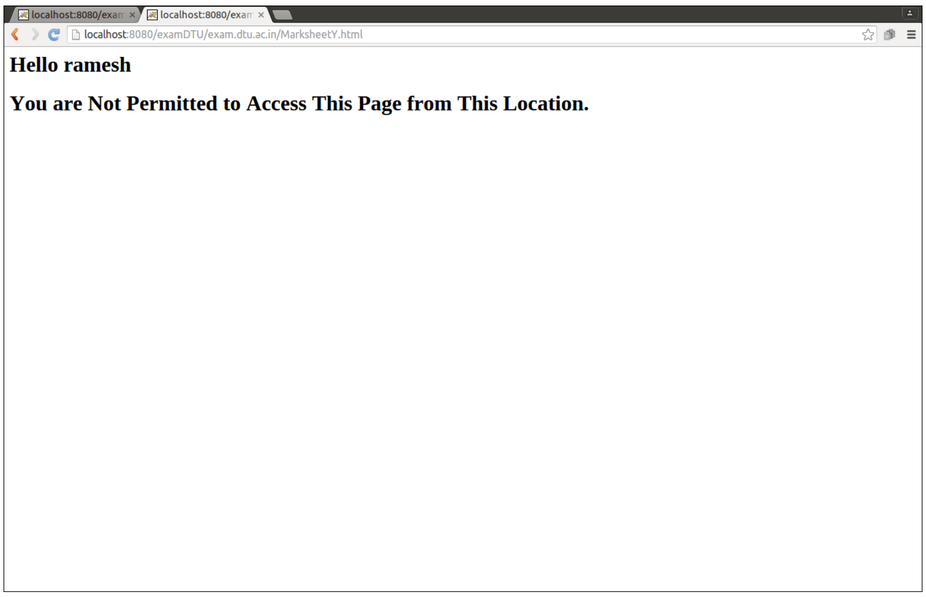
Copyright ? 2011-2012 All rights of the site rest with Delhi Technological University
For Website related queries/suggestions contact examination branch
Sunday 29. May 2016

Figure 4.14: Marksheet Page when Accessed from Particular Location

Location based access control are useful when the organizations want to restrict their information to their internal users or to some other users who belong to some alliance organizations. In order to implement this type of access control, we have used the IP address of the machine from which the request to access a web page is generated. We have inferred the group to which the IP address of the source machine belongs. If that group is an ally then the access is granted otherwise the request is denied.

Even if a user whose role is allows him to access the *marksheet page*, accesses this page from a wrong location then the access is denied. In this case, the context, *i.e.*, `location` plays a key role in deciding the access grant. The page must be accessed from particular allowed locations along with the permissible roles. In that case the access would be granted.

Result of accessing the *marksheet page* from allowed location is shown in figure 4.14. Figure 4.15 shows that access is denied when the page is requested from other locations.


localhost:8080/examDTU/exam.dtu.ac.in/MarksheetY.html

Hello ramesh

You are Not Permitted to Access This Page from This Location.

Figure 4.15: Access Denied to MarkSheet Page

In this chapter, we have described the access control system designed and developed for the examination portal of DTU. We have not only covered the traditional access control based on roles but also covered the access control based on contextual parameters. This work demonstrates the capability of using our proposed framework for practical domains. This is not limited to the academic environment but can be easily extended to other domains.

We also point out and highlight the extensibility of our framework where different parameters can be added as and when needed. This keeps the role management simple and avoids problem like role explosion. The proposed framework deals with many real world policies and paves a way to implement a dynamic and adaptable access control system.

Chapter 5

Conclusion and Future Work

In this thesis, we have shown how the *Access Control* model and *Authorization Policies* can be represented using *Web Ontology Language* (OWL). The importance of this representation is driven from the fact that organizations need access control systems which are dynamic in nature and access grants are decided at run time. These access decisions are mainly guided by the contextual information like *time*, *day*, *location etc.*

We have started with representing the basic access control system for academic domain. We have written ontologies to represent various entities defined in the academic environment. These correspond to the various activities performed by different users in the domain. We have defined different roles, identified conflicts among them and associated permissions with different actions. We have implemented flat, hierarchical and constrained Role Based Access Control (RBAC) system for this example domain.

However, the RBAC model has a specific need to preassign permissions to roles and does not provide an explicit way to include contextual information. We have extended the RBAC system so that different context parameters can be included while deciding the access grant. We modelled *time*, *day*, *location* and *group membership* which are the most practical contextual parameters to help in access control. Various sample high-level policy statements and their corresponding conversion in OWL rules have been shown. The representation is based on a flexible and extensible framework where more parameters can be added as and when the need arises. The high level policy rules are enforced using an inference based reasoning process. We have used the EYE reasoner which has very good performance in comparison to other reasoners.

We have designed and developed a working access control system for the examination portal at DTU. This system demonstrates how the theoretical framework can be extended to meet the need of many organizations. In this system, we have shown enforcement of different high level policy rules with different contextual information. This framework also includes policy administration where existing rules can be modified and new rules can be added. This demonstration shows the particle use of semantic reasoning where different existing facts can be used to derive new information which ultimately helps in access control.

In future, we would like to develop the access control system for different domains including my sponsors. Exploring different access control models, other than classical models, would be an interesting task. Writing more complex policy rules based on wide variety of context would also be considered in the future.

Bibliography

[1] R. Sandhu, D. Ferraiolo, and R. Kuhn, “The nist model for role-based access control: towards a unified standard,” in *Fifth ACM Workshop on Role-Based Access Control*, (Berlin), July 2000.

[2] R. Verborgh and J. D. Roo, “Drawing conclusions from linked data on the web the eye reasoner,” in *IEEE Software*, May-June 2015.

[3] T. B. Lee, “Cwm,” 2000.

[4] “Deep taxonomy benchmark.” `http://eulersharp.sourceforge.net/2003/03swap/dtb-note.`

[5] R. S. Sandhu and P. Samarati, “Access control: principle and practice,” *Communications Magazine, IEEE*, vol. 32, no. 9, pp. 40–48, 1994.

[6] R. S. Sandhu, “Lattice-based access control models,” *Computer*, vol. 26, no. 11, pp. 9–19, 1993.

[7] D. E. Bell and L. J. LaPadula, “Secure computer systems: Mathematical foundations,” tech. rep., The MITRE Corporation, 1973.

[8] J. Park and R. Sandhu, “The ucon abc usage control model,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 128–174, 2004.

[9] X. Jin, R. Krishnan, and R. Sandhu, “A unified attribute-based access control model covering dac, mac and rbac,” in *Proceedings of 26th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2012)*, (Paris, France), July 2012.

[10] S. Godik and T. Moses, "Oasis extensible access control markup language (xacml). oasis committee secification cs-xacml-specification-1.0," November 2002.

[11] M. Dean and G. Schreiber, "Owl web ontology language guide," 2004.

[12] R. Ferrini and E. Bertino, "Supporting rbac with xacml+ owl," in *Proceedings of the 14th ACM symposium on Access control models and technologies*, (Stresa,Italy), pp. 145–154, ACM, June 2009.

[13] J. M. Bradshaw, A. Uszok, M. Breedy, L. Bunch, T. Eskridge, P. Feltovich, M. Johnson, J. Lott, and M. Vignati, "The kaos policy services framework," in *Proc. 8th Cyber Security and Information Intelligence Research Workshop*, 2013.

[14] L. Kagal, "Rei : A policy language for the me-centric project," tech. rep., HP Labs, Sep. 2002.

[15] L. Kagal and T. Berners-Lee, "Rein: Where policies meet rules in the semantic web," *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA*, vol. 2139, 2005.

[16] T. Berners-Lee, D. Connolly, E. Prud'hommeaux, and Y. Scharf, "Experience with n3 rules.," in *Rule Languages for Interoperability*, 2005.

[17] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham, "Rowlbac: Representing role based access control in owl," in *Proceedings of the 13th ACM symposium on Access control models and technologies*, pp. 73–82, June 2008.

[18] "Event ontology." `http://eulersharp.sourceforge.net/2003/03swap/event`.