An RBAC-based Policy Information Base

Timothy E. Squair, Edgard Jamhour, Ricardo C. Nabhen
Pontifícia Universidade Católica do Paraná, PUCPR, PPGIA,
{timothy, jamhour, rcnabhen}@ppgia.pucpr.br

Abstract

This paper presents a framework for representing and distributing access control policies in distributed heterogeneous systems. Access control polices follow the RBAC (Role Based Access Control) model proposed by the NIST. The framework is based on the provisioning strategy defined by IETF, i.e., the RBAC information is represented in terms of a PIB (Policy Information Base) and distributed to the enforcement elements using the COPS-PR protocol. This approach can be explored in several scenarios, for configuring both, network devices and RBAC-aware applications. The provisioning process takes into account the capabilities of the enforcement element, permitting to eliminate or adapt the configuration not supported by the managed device or application.

1. Introduction

This paper presents a framework for distributing access control policies in distributed heterogeneous systems. The framework is inspired on the recently published IETF standards concerning both policy representation and policy distribution, adopting a provisioning approach. The provisioning approach is based on three main elements [2]:

i. a device-independent policy information model, used for representing policies that can be reused among different devices;
ii. a policy information base (PIB), which represents the policy assigned to a specific device. The PIB is generated from the device-independent policy model, by a policy translation process. The translation takes into account the device capabilities, i.e., the mechanisms the specific device supports for enforcing the policy;
iii. a protocol (COPS-PR) [3] specifically designed for supporting policy provisioning using the PIB structure, i.e., negotiating capabilities, transporting and installing the PIB into the device.

The provisioning approach is understandably generic, and can be explored in several management domains. IETF has already explored the provisioning approach for distributing *diffserv* configuration and, recently, for distributing *IPsec* configuration. Other potential target domains for future standardization are MPLS, Access Control and 3GPP UMTS [15]. Each management domain is addressed by defining a device-independent policy information model and a PIB. IETF has published the guidelines for defining these elements. The Policy Core Information Model (PCIM) [5] and its extensions (PCIMe) [6] define a generic set of classes and associations used for representing policies for any management domain. Policy models for specific domains are defined by extending the PCIM/PCIMe structural and association classes. The framework PIB [9] defines a generic PIB template, which specifies the elements required for supporting capability negotiation and policy installation. Again, specific domains are addressed by extending the framework PIB elements.

This paper addresses a domain not yet explored by IETF, i.e., a framework for distributing RBAC (Role Based Access Control) policies to devices and applications. The framework is defined by introducing a device-independent RBAC information model and a RBAC-PIB.

1.1 Motivation

The access control is one of the most important and complex aspects of the security management. The need of access control is present in all the various components of a distributed system. In some cases, the access control refers to the rights to manage network devices, such as gateways and firewalls. In other cases, the access control policies restrict the access of users to shared resources and application level services. RBAC is a quite generic information model that can be used for representing several types of access control policies. In the RBAC NIST model [1], adopted in this work, roles, permissions and resources can be associated in order to express a wide range of access control policies. Additionally, the RBAC NIST model allows to define special rules for constraining undesirable combinations of access permissions using static (session independent) and dynamic (activated in a session) restrictions.

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

In a distributed system, the access control is implemented by different elements. Not all elements support the same enforcement mechanisms neither are capable of interpreting all policy elements defined by the generic RBAC NIST model. The provisioning strategy offers an elegant approach to this problem. The strategy defines (at least) two information model levels: a device-independent information model and another that takes into account the capabilities of the device (PIB). In our work, these models have been called, respectively, RBPIM (Role-Based Policy Information Model) and RBAC-PIB (RBAC Policy Information Base).

The RBPIM allows the RBAC model to be explored from a centralized point of view, allowing the reuse of elements shared by the various devices of the distributed environment (e.g. the role "Network Manager"). Complementarily, the translation process from RBPIM to RBAC-PIB allows the RBAC-policies to be adapted according to the capabilities of the device, by eliminating or adapting the elements not supported by the device.

The use of the protocol COPS-PR also brings evident advantages for distributing policies in a distributed and heterogeneous environment. COPS-PR is specifically designed for supporting policy provisioning using the PIB structure, i.e., negotiating capabilities, transporting and installing the PIB into the devices.

1.2. Contributions

The major contribution of this paper is the RBAC-PIB definition. The RBPIM is based on a previous work described in [12]. In [12], however, the RBPIM model has been explored using the outsourcing approach. Because some extensions have been included in order to support the provisioning approach, the RBPIM is also discussed in this paper. However, it is important to note that, because the PIB is defined by a policy translation, other information models representing RBAC policies could be combined with the PIB strategy.

The remaining of this paper is organized as follows: section 2 reviews some important works that propose alternative approaches for access control policy representation and distribution. Section 3 describes the main elements of the proposed provisioning framework. Section 4 discusses the RBAC-based information model. Section 5 describes the RBAC-PIB. Section 6 presents the performance evaluation of the proposed framework, considering both: the size of the RBAC-PIB and the required time for provisioning the PIB. Finally, the conclusion summarizes the main aspects of this project and points to future works.

2. Related Works

When defining an access control framework two important issues must be considered: (i) the model (or language) adopted for representing policies; (ii) the approach adopted for interpreting, distributing and enforcing the policies. The work described in this paper adopts a PCIM/PCIMe extension for representing RBAC policies and proposes a PIB/COPS-PR approach for distributing the policies. This work also adopts the PDP (Policy Decision Point) and PEP (Policy Enforcement Point) as the entities responsible, respectively, for policy interpretation/distribution and policy enforcement, as defined on IETF [2]. This section will review three other strategies for representing, distributing and enforcing access control policies.

An important work that adopts the PDP/PEP approach for access control is the XACML (eXtensible Access Control Markup Language), proposed by the OASIS consortium [10]. XACML is a complete solution for modeling, storing and distributing descriptive access control policies. The XACML adopts a generic access control model, based on the concept of policies, rules and Targets. A XACML Target is a triple formed by subject, resource and action. Targets are used for selecting which policies must be considered to evaluate a decision request and also for determining if a request is permitted or denied. By properly defining rules and policies, it is possible to model RBAC policies using the "*xacml policy*" language. In fact, the OASIS already published a document supplying the directives for using XACML for describing RBAC policies [11]. XACML-based frameworks are supposed to be implemented using an "outsourcing" PDP/PEP architecture, i.e., policy interpretation is performed by the PDP and final decisions (Permit or Deny) are delivered to the PEPs. A "pure" outsourcing strategy limits the management domains where the XACML framework can be adopted because it lacks a formal method for transmitting configuration to the PEPs [8]. Another important aspect refers to the "reuse" of management information. Policies are described in terms of subjects (e.g., users) and resources (e.g., applications). An important feature for an access control framework is the capacity of reusing management information shared with other management frameworks. In this scenario, describing policies using a standard method for representing management information is a desirable feature. The Common Information Model (CIM) [4] is an important standardization effort for defining a model capable of representing management information. XACML does not directly support the

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 IEEE

creation of policies using CIM elements or any other standard model for reusing management information [8]. By the other hand, PCIMe-based models offer a straight-forward way for creating policies that refers to CIM objects by using "*PolicyExplicitVariables*" (see section 5).

The "Ponder Language" is another important contribution in the policy-based management domain [13]. As opposed to XACML, which addresses only the access control problem, Ponder can be applied to a wide range of management domains. The Ponder language supports distinct types of policies. Authorization policies define the actions that subjects are permitted (or forbidden) to perform on target objects when certain conditions are satisfied. Obligation policies define the actions that subjects must perform on target objects when certain events occur. Composite policies provide facilities for grouping policies and structure them according to the organizational structure or other management needs. The Ponder project is continuously evolving and recent Ponder publications starts exploring the use of CIM as the model for representing the policy target mechanisms and capabilities for *diffserv* frameworks [13]. The strategy adopted for Ponder implementation is quite different from the work described in this paper. The Ponder framework can be implemented by using a toolkit which permits the generation of Java classes for building policy decision and policy enforcement objects. The framework also supports a strategy for notifying events to the policy objects responsible for supporting obligation policies. Our proposal, on the other hand, describes access control policies as a PIB, which is provisioned to the PEP using the COPS-PR protocol. In the PIB, the RBAC policy elements are distinctly identified, offering a flexible method for updating the PIB and notifying information to the PDP. Also, because no assumption is made about the PEP implementation, the RBAC PIB information can be explored by applications or PEPs under distinct strategies.

The work "Role-Based Access Control for XML Enabled Management Gateways" [14] defines a XML/SNMPv3 gateway for RBAC. The RBAC policy is defined in XML terms (using a schema proposed by the authors). The Gateway is responsible for mapping the RBAC XML-policy to a MIB structure and configuring the network devices using SNMPv3. The authors also discuss the advantages of using RBAC policies to simplify the management of network devices. The RBAC-PIB, proposed in this paper, is also represented in XML, but follows the rigid structure defined by the framework PIB [9]. However, a similar approach as described in [14] could be used for applying the RBAC policy to network devices, i.e., create a RBAC-PIB/SNMPv3 gateway. Conceptually, this gateway could be considered as part of a PEP.

There are also tool kits available for simplifying the process of building COPS-PR based frameworks [15]. The white paper [15] also presents an interesting discussion about the advantages of applying the provisioning approach for several management domains, including access control.

3. The RBAC Provisioning Framework

Fig. 1 illustrates the three main elements in the RBAC provisioning framework: the policy server, the policy client and the policy repository (LDAP server). The policy server (i.e., the PDP) is the entity responsible for interpreting and distributing the policy information to the policy clients. A PEP can be considered the component in the policy client responsible for communicating with the PDP and installing the configuration into the device. The communication between the PEP and the PDP is implemented by the standard COPS-PR protocol. The PDP and the PEP keep a permanent TCP connection. This feature permits to the PDP to update the PEP's configuration at any time. Fig. 1 illustrates only one PEP, but a single PDP in the provisioning approach can handle a large number of PEPs. The performance issue is addressed in section 6. As suggested by IETF, the policy and CIM information are both mapped to an LDAP schema. Because LDAP supports remote references through its schema, the CIM and LDAP repository are not required to be implemented in the same LDAP server. Also, both, policy and CIM information could be implemented using other technology, such as XML.

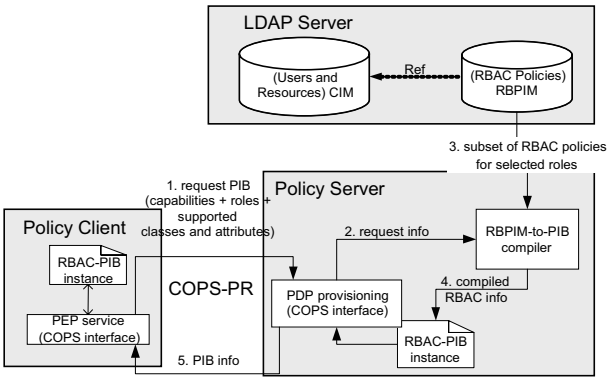

Figure 1. Framework Overview

The RBAC-PIB information can be explored by two approaches, as illustrated in Fig. 2. In the first approach (A), a PEP represents a server application which can be responsible for serving a large number of

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

clients. The communication protocol between the server application and its clients is not imposed by our framework. In our current implementation, the server application communicates with the RBAC framework through a set of RBAC-based API, which follows the definitions proposed by the NIST standard [1]. These API are described in details in [12]. In the second approach (B) the information in the RBAC-PIB is translated to configuration commands to the underlying system or to network devices via SNMPv3 or CLI – Command Line Interface.

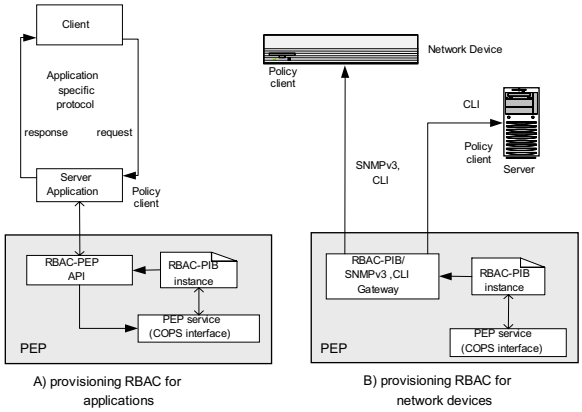

Figure 2. Exploring the PIB Information

The typical sequence of events related to policy provisioning is illustrated in Fig. 1 (the explanation in this section follows the numbers on the arrows in the figure). When initialized, the PEP establishes a COPS-PR connection to the PDP, and requests an initial policy provisioning (i.e., a "full state" request) (1). As defined by IETF, the PEP supplies in the policy request message a combination of "roles+capabilities" that are used to select a sub-set of policies that are required by the application(s) or device(s) interface(s) the PEP represents (e.g., "Warehouse Server" or "DMZ firewall Inbound Interface"). On receiving the request, the PDP activates the RBPIM-to-PIB compiler in order to generate an RBAC-PIB for the PEP (2). The RBPIM-to-PIB compiler collects the subset or RBAC policies associated with the selected "roles" (3) and compiles the information into an RBAC-PIB (4). The RBPIM-to-PIB transformation is described in section 5. The PDP returns the PIB information to the PEP, using the COPS-PR protocol (5). The PEP stores the PIB information in a local repository. The PDP also keeps a copy of the RBAC-PIB in memory. This is required, because the COPS-PR is a stateful protocol, and the PIB information is required in order to restore or updated the PEP configuration.

4. RBPIM

This section discusses the RBPIM model. As defined in [1], the RBAC model includes sets of five basic data elements called users (USER), roles (ROLES), objects (OBS), operations (OPS), and permissions (PRMS). The main idea behind the RBAC model is that permissions are assigned to roles instead of being assigned to users. The User Assignment (UA) is a many-to-many relationship (i.e., a user can be assigned to one or more roles, and a role can be assigned to one or more users). The Permission Assignment (PA) is also a many-to-many relationship (i.e., a permission can be assigned to one or more roles, and a role can be assigned to one or more permissions). A permission is an approval to perform an operation (e.g., read, write, execute, etc.) on one or more RBAC protected objects (e.g., a file, directory entry, software application, etc.). Role hierarchies define an inheritance relation of permissions among roles. The Static Separation of Duty (SSD) model element introduces static constraints to the User Assignment (UA) relationship by excluding the possibility of the user to assume conflicting roles. An important concept in RBAC is that roles must be activated in a session. The Dynamic Separation of Duty (DSD) model element introduces constraints on the roles a user can activate within a session.

The RBPIM (Role-Based Policy Information Model) is a PCIM extension for describing access control policies based on RBAC. RBPIM adopts the RBAC model described in [1], but extensions have been introduced in order to provide a more flexible method for mapping users to roles and describing permissions, and also for establishing network topology-based and time-based permission constraints. Fig. 3 shows the revised RBPIM model adapted to the provisioning approach. The gray classes were introduced by the RBPIM model. The others are defined by PCIM/PCIMe [5,6] and CIM Core[4]. Table 1 presents a short description of the classes used in the RBPIM.

The *RBACPolicyGroup* defines a set of policy information that must be considered when generating a *RBAC-PIB*. Usually, in a large distributed environment, the policy repository will contain a large number of *RBACPolicyGroup* instances, each one associated with one or more *PolicyRoleCollection* instances. When a PEP requests the policy provisioning from the PDP, it supplies the "roles" assigned to its interface(s). By using the *PolicySetInRoleCollection* association, the PDP selects only the *RBACPolicyGroup* instances that must be considered for that particular interface.

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

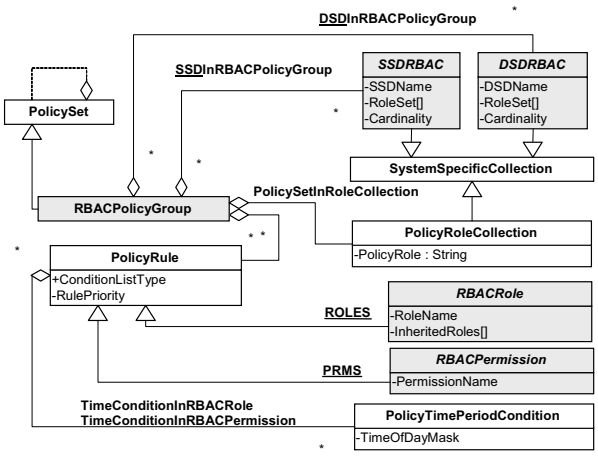

Figure 3. RBPIM Classes and Associations

Table 1. RBPIM Classes

RBPIM	Class Description
RBACPolicyGroup	Defines which RBAC policy elements must be considered in order to generate a PIB for a specific device interface.
RBACRole	Represents roles ∈ ROLES
- UACompoundCondition	Defines a logical expression that permits to select a set of users (represented by CIM objects)
- AssignerRBACPermission	Defines one or more labels that represent the permissions assigned to a role. The labels correspond to *RBACPermission* instances.
RBACPermission	Represents permissions ∈ PRMS.
-OBSCompoundCondition	Defines a logical expression that permits to select a set of objects ∈ OBS (represented by CIM objects)
-AssignerOperation	Defines a multiple-value string field representing a set of operations ∈ OPS that can be performed on CIM objects.
-PacketFilterCondition	Defines a set of IP header filters used to restrict the permissions according to the network topology.
PolicyTimePeriodCond.	Defines a set of time conditions used to impose restrictions to role and permission Assignments.
SSDRBAC	Defines a set of roles and cardinality, used to impose restrictions on the roles that can be assigned to the same user.
DSDRBAC	Defines a set of roles and cardinality, used to impose restrictions on the roles that can be simultaneously activated by a user in a session.

PCIM defines that a Policy is defined by a set of *PolicyRules*. A *PolicyRule* represents the following expression: "*If conditions are satisfied*, *then apply actions*". The conditions are expressed by a logical expression that combines attributes of objects already described in the CIM repository. In order to provide a flexible method for defining user assingments and representing permissions, RBPIM introduces two *PolicyRule* extensions, named *RBACRole* and *RBACPermission*.

An *RBACRole* instance express the following rule: "*If the conditions imposed to the user(s) attributes are satisfied then the role and the corresponding permissions can be assigned to the user*". An *RBACPermission* instance express the following rule: "*If the conditions imposed to the object(s) attributes are satisfied then the operation(s) can be performed on the object(s)*". The *RBACRole* class and its associations are illustrated by Fig. 4. The *RBACPermission* class and its associations are illustrated by Fig. 5. As shown in the figures, RBPIM introduces two *CompoundPolicyConditions* extensions: *UACompoundCondition* and *OBSCompoundCondition*. These classes define, respectively, a reusable set of users and a reusable set of objects that can be shared by distinct rules.

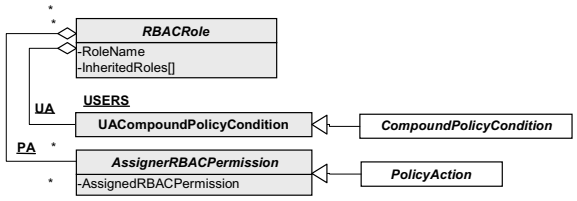

Figure 4. RBACRole

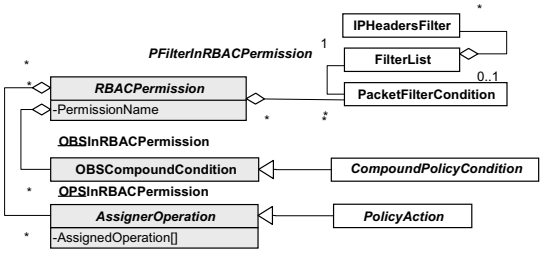

Figure 5. RBACPermission

Note that because PCIM does not permit a direct association between PolicyRule instances, *RBACRole* and *RBACPermission* are indirectly associated by matching the attribute *AssignedRBACPermission* (from *AssignerRBACPermission*) with the attribute *PermissionName* (from *RBACPermission*).

The static and dynamic separation of duty constraints are represented, respectively, by DSDRBAC and SSDRBAC instances, according to the semantic described in [1]. Both classes are specializations of "*SystemSpecificCollection*", defined

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

by the CIM Core. Note that the SSD and DSD constraints are imposed on the *RBACPolicyGroup*. Therefore, they could not be represented as rule conditions.

4.2. Example

The examples in Fig. 6 and 7 illustrate the use of the RBPIM model. The *RBACRole* in the figure was called "Auditor". The attribute *InheritedRoles* is used for expressing the Hierarchical RBAC, i.e., the role "Auditor" inherits the permissions of role "Employee". The UA relationship for "Auditor" points to a compound condition with a single simple condition, based on a *PolicyExplicitVariable*. The explicit variable permits to create conditions referring to CIM objects. In this case, the "Auditor" role is assigned to all users where the *BusinessCategory* attribute match "C1". The UA assignment is restricted to the period between 10h00 and 16h100 by the *PolicyTimePeriodCondition* instance.

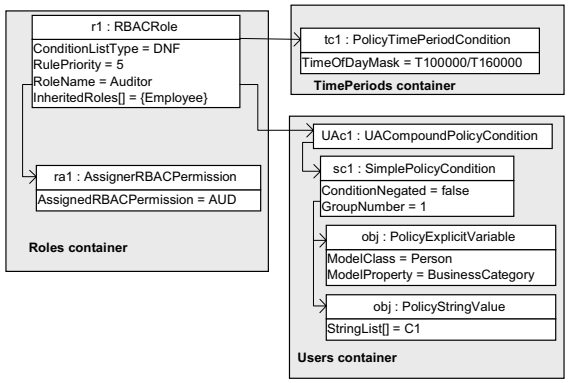

Figure 6. RBPIM RBACRole example

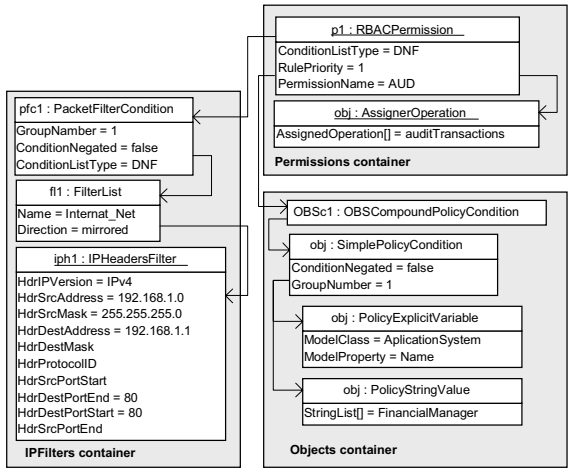

Figure 7. RBPIM RBACPermission example

The "Auditor" has a PA relationship with a permission called "AUD". This permission defines that the operation "*auditTransactions*" can be executed when *OBSCompoundConditions* and the *PacketFilterCondition* are simultaneously satisfied. In this case, the compound condition includes an explicit variable condition pointing to a CIM object that represents a specific application. The *PacketFilterCondition* restricts the operation from machines within the 192.168.1.0/24 subnet.

As well as PCIM, the RBPIM model is implementation neutral. RBPIM mapping to LDAP schema has been implemented according to the IETF standard PCLS [7]. In Fig. 6 and 7, we define six containers where the policy objects are stored. In our approach, a container is created for storing "reusable" information. For example, "*RBACRole*" instances are stored in a "Roles" container. *AssignerRBACPermission* instances, on the other hand, are too simple to justify a container, because rewriting its single attribute is cheaper than creating a reusable object and pointing to it. Hence, they are also stored in the "Roles" container and associated to *RBACRole* instances by DIT containment. *UACompoundPolicyCondition* instances are also worthy of being considered reusable information and, therefore, are stored in a specific container. *RBACRoles* instances have received the required attributes for pointing to the *UACompoundCondition* instances and grouping then according to a DNF or CNF strategy. The same reasoning applies to the other classes in the figure, i.e., that needs to be reusable information receives a container and other associations are implemented by DIT containment.

5. RBAC-PIB

This work defines a RBAC-PIB which represents the information transferred from the PDP to the PEP during the provisioning process. The RBAC-PIB is based on the IETF framework PIB definitions [9]. The framework defines a generic PIB, which include classes for supporting information used by the COPS-PR protocol. Specialized PIBs are defining by including additional classes for representing the information related to a particular domain.

Fig. 8 shows the RBAC-PIB structure represented in XML. A PIB can be described as a conceptual tree namespace where the branches of the tree represent structures of data or Provisioning Classes (PRCs), while the leaves represent various instantiations of Provisioning Instances (PRIs). The PRCs corresponding to the *BasePib*, *DeviceCapabilities* and *ClassifierGroup* groups are defined by the Framework

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

PIB [9]. All PIB elements corresponding to branches have an "oid" attribute defined according to RFC 3159. The "oid" prefix 1.3.6.1.2.2.2 refers to the framework PIB definition. The PRCs corresponding to the RBAC group are extensions defined by our proposal. The oid prefix 1.3.6.1.2.2.2.6, currently unused, has been assigned to identify the PRC classes corresponding to the RBAC information.

A PRC can also be used for both, supplying ("*notify*") or receiving ("*install*") information to/from the PDP. A PRC can be also "*install-notify*", providing a bidirectional exchange of information between the PEP and the PDP. The *BasePib* has three tables, grouping the instances of the PRCs named *PrcSupport*, *PibIncarnation* and *DeviceId*. All classes are "notify", except *PibIncartion* which is "install-notify". The *PrcSupport* instances define the classes and attributes supported by the PIB. As example, Fig. 8 shows the PRI (instance with id="9") corresponding to the PRC *User* (in the *UserAssignment* element of the RBAC group). The *SupportedAttrs* attribute is a binary map indicating the PEP supports all three attributes defined for the class. This information is used by the PDP, in order to determine which attributes must be transferred to the PEP during the provisioning process.

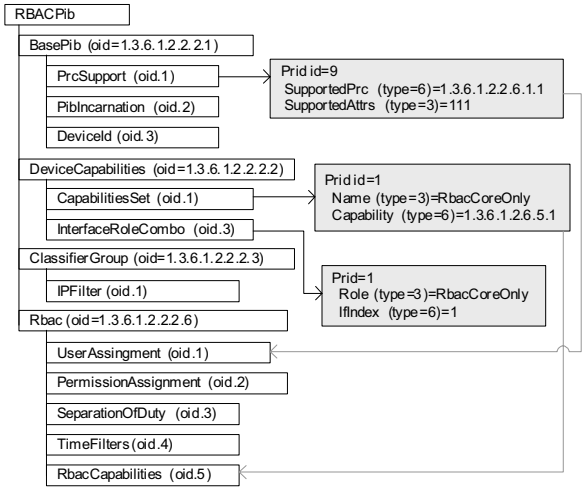

Figure 8. RBAC-PIB structure

The *PibIncarnation* instance (this PRC contains exactly one row) includes information about the PDP, the version of the policy currently downloaded and the behavior of the PEP when the connection with the PDP is closed. An attribute, called *<FullState>*, plays an important role in the provisioning process. The PEP use *FullState=true* to ask for a full state update to the PDP (in this case, any previous state in the PDP is erased) and *FullState=false* for asking an incremental update. *DeviceId* supplies additional information for the PDP to identify the PEP (e.g., RBAC version or model).

DeviceCapabilities group supplies information to the PDP permitting it to select and adapt the policies to be provisioned to the PEP. The *CapabilitiesSet* defines pointers to specific capabilities defined by the *RbacCapabilities* section in the *RbacGroup* (explained further in this section). The *InterfaceRoleCombo* instances indicate the roles and capability sets that have been assigned to each interface of the managed element.

The *Classifier* contains the PRC *<IPFilter>*, permitting the description of filtering conditions based on the fields of the IP header. This PRC is used to represent the *IPHeadersFilter* conditions used in the RBPIM model for constraining the PA assignments.

Table 2. Structural Classes Mapping

RBPIM	RBAC PIB *PRC*	PIB Group
RBACRole	*Roles*	*Rbac User Assign.*
UAComp.PolicyCond.	*Users*	*Rbac User Assign.*
AssignerRBACPerm.	*Permission attribute in RolePermissions PRC*	*Rbac Perm. Assign.*
RBACPermission	*Permissions*	*Rbac Perm. Assign.*
OBSCompoundCond.	*Objects*	*Rbac Perm. Assign.*
AssignerOperation	*operation attribute in Permissions PRC*	*Rbac Perm. Assign.*
SSDRBAC	*Not Mapped*	
DSDRBAC	*DSD*	*Rbac Sep. of Duty*
PolicyTimePeriodCond.	*TimeFilters*	*Rbac TimeFilters*
PacketFilterCondition	*IPFilter*	*Classifier*

Table 3. Association Classes Mapping

RBPIM	RBAC PIB PRC	RBAC PIB Group
UA	*UserRoles*	*Rbac User Assign.*
TimeCondInRBACRole	*RoleTimeFilters*	*Rbac User Assign.*
PA	*RolePermissions*	*Rbac Perm. Assign.*
TimeCondInRBACPerm.	*Perm.TimeFilters*	*Rbac Perm. Assign.*
PF.CondInRBACPerm.	*Perm.IPH.Filters*	*Rbac Perm. Assign.*
OBSInRBACPermission	*object attribute in Permissions PRC*	*Rbac Perm. Assign.*
OPSInRBACPermission	*operation attribute in Permissions PRC*	*Rbac Perm. Assign.*
DSDInRBACPolicyGr.	*DSDEntries*	*Rbac. Sep. of Duty*
SSDInRBACPolicyGr.	*Not Mapped*	

The strategy adopted for defining the representation of the RBAC information follows the framework PIB definitions. PRC classes are used to group the RBAC information, all attributes are defined within PRI instances and pointers based on "oid's" are used to implement the association between classes. This approach is required in order to use the COPS-PR protocol for provisioning the policy information to the PEP. According to our proposal the RBAC group contains five elements: *<UserAssignment>*,

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

<*PermissionAssignment*>, <*SeparationOfDuty*>, <*TimeFilters*> and <*RbacCapabilities*>.

Table 2 summarizes the mapping between the RBPIM structural classes and RBAC PIB PRCs. Table 3 summarizes the mapping between the RBPIM association classes and RBAC PIB PRCs. The PIB PRCs are explained further in this section.

Fig. 9 shows the structure of the <*UserAssignment*> element of the <*Rbac*> group. The PRIs in the figure correspond to the policy example described in Fig. 6 and 7. The <*UserAssignment*> element contains four PRCs: <*Users*>, <*Roles*>, <*UserRoles*> and <*RoleTimeFilters*>. Each PRI in the <*Users*> PRC corresponds to a user identified by the <*uid*> attribute. The <*pwd*> and <*pwdmeth*> attributes are optional (i.e., as informed by the <*PrcSupport*> structure). The authentication attributes are optional, because authentication management can be outside of the framework scope.

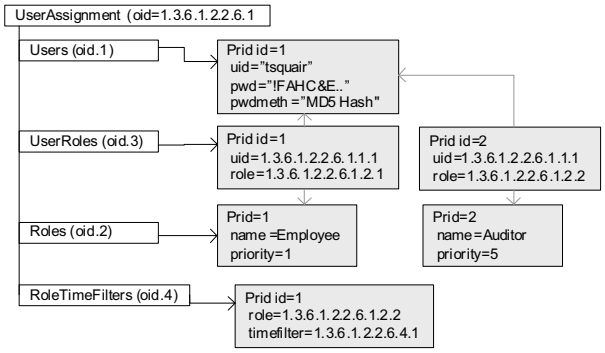

Figure 9. RBAC PIB: UserAssignement Group

Similarly, each PRI in <*Roles*> corresponds to a RBAC role as defined by the *RBACRole* class in the RBPIM model. The UA assignment is defined by the <*UserRoles*> PRC, which is an association class between <*User*> and <*Roles*>. Note that all *UACompoundPolicyCondition* information in the RBPIM model is pre-processed by the PDP and the result is expressed by the <*UserRoles*> instances. During the process of defining the <*UserRoles*> PRIs, the PDP automatically creates the PRIs for representing the roles indirectly assigned to a user by heritage (by the *RBACRole.inheritedRoles[]* attribute in the RBPIM model). During the process of defining <*UserRoles*>, the PDP also takes into account the SSD constraints (corresponding to the *SSDRBAC* class in the RBPIM model), which results in only the highest priority roles free of SSD constraints are assigned to a user in the PIB. Finally, the <*RoleTimeFilters*> is used to constraint the period a user can activate a role. Note that the time filter information refers to the <*TimeFilters*> PRC in the <*Rbac*> group structure in Fig. 8. <*TimeFilters*> PRIs are reusable constraints directly translated from the *PolicyTimePeriodCondition*. These time-constraints can also be referred by the <*PermissionTimeFilters*> PRC .

Fig. 10 shows the structure of the <*PermissionAssignment*> element of the <*Rbac*> group. The <*PermissionAssignment*> element contains five PRCs: <*Objects*>, <*Permissions*>, <*RolePermissions*>, <*PermissionsIPHeadersFilters*> and <*PermissionTimeFilters*>. The <*Objects*> PRC defines the resources controlled by the RBAC policy. The resources are represented by CIM objects. Each <*Objects*> PRI contains a Boolean expression, formed by grouping policy explicit variables in CNF or DNF form. The <*Permissions*> PRC defines permissions by mapping an operation (defined as a string attribute) to an <*Objects*> PRI. Alternatively, CIM also offers elements for describing standard operations. In this approach, the operation is included in the <*Objects*> expression. <*RolePermissions*> is the association class responsible for assigning permissions to RBAC roles. Permissions are constrained by the PRCs <*PermissionIPHeadersFilters*> and <*PermissionsTimeFilters*>, permitting to define, respectively, subnet constraints and time period constraints to the permissions assigned to a role. Note that <*PermissionIPHeadersFilters*> employees "oid" references to the <*IPFilter*> element defined by the Framework PIB.

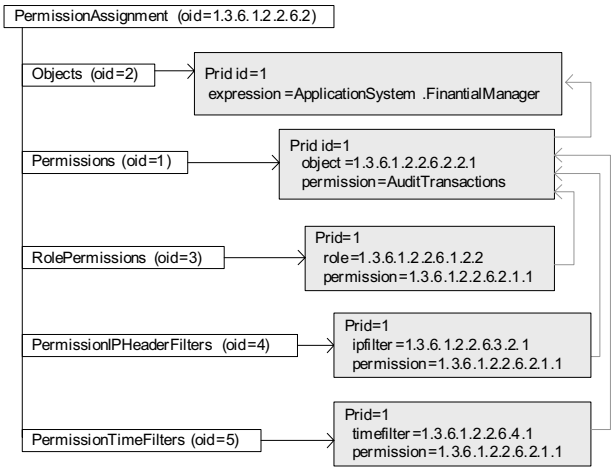

Figure 10. RBAC PIB: Permission Assignment

The <*SeparationOfDuty*> element (see Fig. 11) contains the RBAC definitions permitting the PEP to implement the dynamic separation of duty, i.e., constraints the roles a user can simultaneously activate

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

within a section. The <*DSD*> PRC defines the DSD cardinality, and the <*DSDEntries*> PRC defines the roles constrained by the DSD. Note that the "static separation of duty" constraints are pre-processed by the PDP, and therefore, are not included in the PIB.

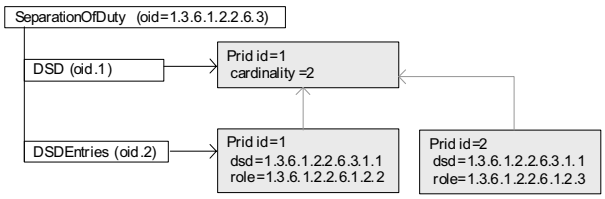

Figure 11. RBAC PIB: Separation of Duty

<*RbacCapabilities*> contains the elements pointed by the <*CapabilitiesSet*> from the framework PIB. It is composed by 5 elements (see Fig. 12). <*RbacCoreCaps*> is a mandatory capability that defines the support to the basic access control functionalities, as defined by the NIST. Presently, only the NIST model is supported, but future extensions could include alternative models. <*RbacDSDCaps*> defines the support for dynamic separation of duty constraints. Usually, network devices have no support to this functionality. <*RbacIPFilterCaps*> and <*RBacTimeFilterCaps*> define the support for network and time constraints imposed to *Rbac* permissions and roles activation. These features are not presented in the NIST specification, being extensions proposed by the RBPIM.

Finally, <*RbacIncrementalUploadCaps*> defines an optional framework feature. When this feature is present, the PIB information generated in the initial provisioning process is not complete, because it lacks the UA assignment (i.e., the mapping between user and roles). The UA assignment is not initially provisioned because an application with a large number of potential users would lead to an extremely large PIB. Instead, the UA assignment is incrementally uploaded to the PIB when a new RBAC session is created. The PEP requests the UA assignment to the PDP using the COPS-PR protocol (*FullState*=false) and receives only the PIB elements concerning the UA assignment of the new user. After this event, the check access requests can be locally decided by the policy client. Note that this feature is not useful when the PIB is used for generating configuration commands (the traditional PIB approach); it applies only to RBAC-aware applications.

In our current implementation, when the *IPFilter* capabilities are not supported, the corresponding *Permission* PRIs are simply eliminated. Because our model defines only "positive" actions, eliminating permission rules will never result in additional rights being assigned to a role. If the DSD capability is absent, the PDP process the DSD as SSD constraints (i.e., only higher priority roles free of DSD and SSD are provisioned to the PEP). This approach can considerably reduce the permissions assigned to the PEP, but it is a better approach than simply eliminating all roles in the *RBACPolicyGroup*. Also, the *TimeFilter* capability absence does not require eliminating all corresponding *Permissions* and *Roles*. Instead, using the COPS-PR permanent connection, the PDP can dynamically perform the PIB actualization of the PEPs in order to replace the policy elements affected by the time constraints.

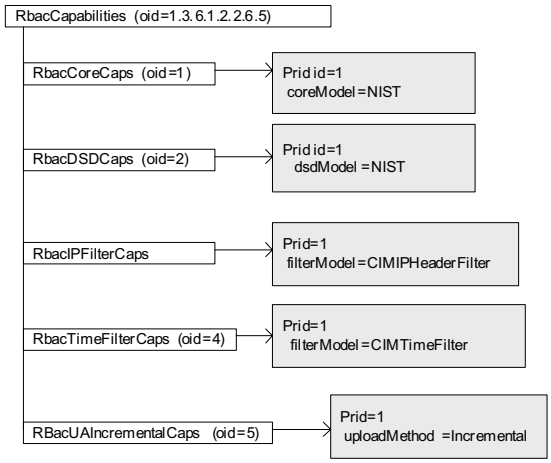

Figure 12. RBAC Capabilities

6. Evaluation

Our proposal has been evaluated in terms of two criteria: a) The size of the PIB with respect to the complexity of the RBAC policy, i.e., the number of policy elements (Roles, PRMS, OBJS, OPS, DSD and SSD, as defined in session 4) that must be processed in order to generate the PIB; b) The time required for provisioning a PIB. The provisioning time includes the time for compiling the RBPIM model, generating the PIB and transferring it to the PEP using the COPS-PR protocol.

The PDP has been implemented in Java and runs in a Pentium IV 1.6 GHz, 1GB memory PC. The policies are stored in an *OPenLDAP* server, version 2.7. The PDP/PEP is connected by a 100 Mbps Ethernet LAN. The evaluated scenario corresponds to provisioning RBAC for applications (see item *A* in Fig. 2). In this scenario, we assume a managed device with support to the <*RbacIncrementalUploadCaps*), i.e., user information is not initially provisioned. Table 3

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**

summarizes the results of the provisioning time evaluation for a subset of RBAC policies that concerns "a single PIB" (as defined by the interface roles of the managed element). Note the effect of the number of RBAC objects in the PIB size and provisioning time. The time for provisioning "one" user is also presented in the table. The PIB size corresponds to the XML representation adopted in this paper.

Table 3. Evaluation of Provisioning Time

Roles	PRMS	OBJS	OPS	DS	SSD	Full (ms)	1 User (ms)	Size (kb)
10	6	3	8	1	3	2,1	0,28	16
40	6	3	8	1	3	3,8	0,51	34
80	6	3	8	1	3	6,5	1,1	58
20	10	7	12	1	3	3,0	0,37	24
20	40	37	42	1	3	4,3	0,44	42
20	80	77	82	1	3	7,2	0,41	67
20	6	3	8	10	3	2,8	0,45	25
20	6	3	8	40	3	2,9	0,39	36
20	6	3	8	80	3	3,7	0,43	51
20	6	3	8	1	10	2,9	0,42	22
20	6	3	8	1	40	2,7	0,44	22
20	6	3	8	1	80	2,7	0,41	22

7. Conclusion

This paper has presented a policy based framework for deploying RBAC policies in heterogeneous and distributed systems adopting a provisioning approach. The framework has been implemented in accordance with the IETF standards and a superset of the NIST RBAC standard. This work has proposed a RBAC-based information model and a RBAC-based PIB. The development of this work has shown that specialized PIBs can be easily created by extending the framework PIB. Also, the capabilities concept is very useful for creating policies for heterogeneous systems. This concept has been very useful for deploying RBAC, which is a complex model with many optional features. The COPS-PR protocol has been equally very useful for developing a method for installing and updating the RBAC configuration without overloading the device/application with unnecessary configuration. Future works include extending the provisioning approach for other access control languages, and building a SNMPv3 gateway for the RBAC PIB.

8. References

[1] D.F. Ferraiolo, R.S. Sandhu, G. Serban; "A Proposed Standard for Role-Based Access Control", *ACM Transactions on Information System Security*, Vol. 4, No. 3, (2001) pp. 224-274".

[2] J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser; "Terminology for Policy-Based Management", IETF RFC 3198, Nov. 2001.

[3] Chan K.; Seligson, J.; Durham, D.; Gai, S.; McCloghrie, K.; Herzog, S.; Reichmeyer, F.; Yavatkar, R.; Smith, A.; "COPS Usage for Policy Provisioning (COPS-PR)", IETF RFC 3084, Mar. 2001.

[4] Distributed Management Task Force (DMTF); "Common Information Model (CIM) Specification", URL: http://www.dmtf.org (2003).

[5] B. Moore, E. Elleson, J. Strasser, A. Weterinen; "Policy Core Information Model", IETF RFC 3060, Feb. 2001.

[6] B. Moore, E. Elleson, J. Strasser, A. Weterinen; "Policy Core Information Model Extensions"; IETF RFC 3460, Feb. 2001.

[7] J. Strassner, E. Ellesson, B. Moore, R. Moats. "Policy Core Lightweight Directory Access Protocol (LDAP) Schema", IETF RFC 3707, Feb. 2004.

[8] Toktar, Emir , Jamhour, E., Maziero, Carlos. "RSVP Policy Control using XACML", *IEEE 5th International Workshop on Policies for Distributed Systems and Networks. (POLICY 2004)*, New York, 2004, pp. 87-98.

[9] R. Sahita, S. Hahn, K. Chan, K. McCloghrie; "Framework Policy Information Base", IETF RFC 3318, Mar. 2003.

[10] OASIS: "eXtensible Access Control Markup Language (XACML)", version 1.03. OASIS Standard, Feb. 2003, URL: http://www.oasis-open.org

[11]. OASIS; "XACML Profile for Role Based Access Control (RBAC)", draft, Feb. 2004, URL: http://www.oasis-open.org

[12] R. Nabhen, E. Jamhour, C. Maziero; "Policy-Based Framework for RBAC", *14th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, Germany (DSOM 2003),* Oct. 2003, pp. 181-193.

[13] L. Lymberopoulos, E. C. Lupu, M. S. Sloman, "Ponder Policy Implementation and Validation in a CIM and Differentiated Services Framework", *NOMS 2004*, Seoul, Apr. 2004.

[14] V. Cridlig, O.Festor, R.State, "Role-Based Access control for XML Enabled Management Gateways", *15th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management* (DSOM 20040, Davis, USA, Nov. 2004, pp. 183-195.

[15] R. Fenger, H. Hegde, D. Larson, R. Sahita, "Simplifying Support of New Network Services Using COPS-PR", *Intel Corporation white paper*, 2002, URL http://www.intel.com/labs/manage.cops.

Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)
0-7695-2265-3/05 $20.00 © 2005 **IEEE**