Policies in SNMPv3-based Management

Salima Omari	*Raouf Boutaba*	*Omar Cherkaoui*
Laboratoire PRiSM	*ECE Department*	*Lab téléinformatique*
Université de Versailles	*University of Toronto*	*Université UQAM*
Versailles, FRANCE	*Toronto, ON, CANADA*	*Montréal, CANADA*
osa@prism.uvsq.fr	*rboutaba@comm.utoronto.ca*	*cherkaoui.omar@uqam.fr*

Abstract

Two important achievements in the network management area motivated the work presented in this paper. The first one is the wide acceptance of the policy concept and its introduction as a means for driving management procedures. The second concerns the capabilities brought by the version 3 of the SNMP protocol for configurable and secure network management. The deployment of SNMPv3 at equipment level allows henceforth concretizing the policy-driven management: Refining enterprise policies; and enforcing them down the managed network resources. This paper aims at integrating the policy concept into the SNMPv3 framework. It proposes a set of rules to map authorization policies to the VACM (View Based Access Control Model) standardized as part of the SNMPv3 management framework. Policy attributes are maintained in a configuration database local to the SNMPv3 entity and a new application is incorporated into the SNMPv3 entity to perform the mapping. This will ultimately allow manager and management applications to enforce enterprise authorization policies independently of the security model(s) implemented by SNMPv3 entities.

Keywords

SNMPv3, security, authorization policies, view-based access control.

1. Introduction

The heterogeneity of network equipment in a multi-vendor environment has been, and is still, a problem faced by network managers and management application developers. Large efforts have been dedicated during the 20 last years to solve the heterogeneity problem by specifying standard management interfaces and protocols. However, these efforts provided by a number of standardization bodies, forums and consortia led to the emergence of multiple standard management protocols and to the proliferation of network management models. The coexistence of different management standards raised a new heterogeneity problem. This problem is tackled in practice by developing appropriate gateways, which ensure the mapping between information models and the conversion of management protocols.

The Simple Network Management Protocol (SNMP) is the most widely deployed standard protocol for managing network devices. However, the simplicity of the protocol is obtained at the expense of a lack of functionality, which is becoming more

(c) 1999 IFIP

apparent with the rapid growth of current networking environments in terms of size and number of resources attached to them. The latest version of the protocol, SNMP version 3 [8], has been adopted within the IETF (Internet Engineering Task Force) and enhances the capabilities of the protocol particularly in terms of security. Data integrity, authentication, privacy and access control are supported by SNMPv3. These new security features make SNMP-based management more reliable, and hence, ready to be adopted for security-sensitive enterprise network management.

In addition, to deal with the complexity due to the large size of current networking environments, the automation of management processes is increasingly demanded. Indeed, in such environments, human managers cannot handle the large number and the variety of resources to be managed unless appropriate and automated tools are made available to them. The policy concept has been introduced as a means to capture enterprise requirements, derive management plans and drive their execution in the network. Policies are defined as the plans of an organization to achieve its management goals [1]. They allow to reduce the gap between high-level abstract enterprise management goals and executable control actions and to help during the process of making management decisions [2]. Policy-based management is now widely adopted by network and systems management communities ([1-6]). Traditionally, high-level management policies are written on scratch pads, saved on E-mail, or just noted mentally. Low level management policies are hard-coded into management applications and are hence difficult to modify to adapt to changes in network infrastructure or to enterprise goals. Recent policy-based management approaches specify management policies as managed objects, which can be dynamically initialized, modified and removed, this way facilitating the evolution of management applications.

The objective of this work is to support policy-based management within the standard SNMPv3 management framework. This is achieved by incorporating a policy enforcement application into the SNMPv3 entities. This application essentially performs a mapping between policy object attributes and the information model implemented by the entity in its local configuration database. This paper focuses on the mapping of authorization policies ([1]) into the access control model supported by the SNMPv3 entity. The implementation particularly considers the View-based Access Control Model (VACM) [10] standardized within the SNMPv3 framework. It is done as part of ModularSNMPv3 project at University of Quebec At Montreal (UQAM), which essentially implements a working Modular SNMPv3 engine in Java.

The access control subsystem part of the standard SNMPv3 framework is designed so as to support different access control models such as the VACM or others. Incorporating the policy enforcement service in the SNMPv3 entity allows abstracting the details of the supported access control model from management applications. This will allow for automation, greater flexibility and non-conflicting network-level management. Indeed, the (re-)configuration of access control rules will be performed by management applications manipulating authorization policies and reflected at the level of individual network entities by the corresponding mapping service. The latter will automatically map the authorization policies to MIB security parameters according to the supported access control model and transparently to management applications.

(c) 1999 IFIP

The paper is organized as follows. Section 2 presents an overview of the policy concept for network and distributed systems management. Section 3 introduces the new security features of the standard SNMPv3 framework. Section 4 introduces the rules defined for the mapping of authorization policies into the standard View-based Access Control Model. In section 5, we describe the implementation of the mapping service and the policy enforcement application within the overall ModularSNMPv3 architecture. Finally section 6 concludes the paper.

2. The Policy concept

In the context of integrated network and systems management, policies define technical management measures that are specific to a particular set of managed objects. In the context of an organization, policies are guidelines and plans to achieve the organization goals. There is no standard definition for designating management policies. In general, policies are used whenever goals, rules, laws, tactics or strategies for the management process need to be expressed. Automation of the management process can be achieved using policies, which may help in making management decisions to respond to particular situations observed in the network, and/or specify the control plans to be followed in order to achieve specific goals. The followings are typical examples of management policies:
Policy_1: Back up file servers every Friday.
Policy_2: Inventory network resources twice a year.
Policy_3: Customer may access the service if she/he has subscribed to it.
Policy_4: Service provider/manager has to maintain the QoS as contracted to users.

The previous policy examples are defined at different abstraction levels. Indeed, policies can be applied from higher level to lower level management. High-level policies describe roughly how a manager will reach a given goal, while low-level policies describe precisely what control actions should be performed on the managed resources. A process is needed to refine high level abstract policies and derive policies that are more specific to the managed network resources and which should be easily enforced in the system.

Large networked systems are composed of a large number of resources. Therefore, it is unreasonable to specify a policy for each managed resource. Domains have introduced as a means to group resources for management purposes and to distribute management responsibilities [1], [2]. The grouping of objects into domains may be done according to their functionality, their physical location, etc. A domain may be composed of a set of managing objects, which apply the same management policy or a set of managed objects to which a given policy applies [1]. Domains can also be more sophisticated structures grouping both managed and managing objects [2].

Four types of policies have been identified [1]: positive authorization policies; negative authorization policies; positive obligation policies; and negative obligation policies. The two first types represent what is allowed (respectively not allowed) to be done on the managed objects. The two last types represent what must (respectively must not) be done on the managed objects.

(c) 1999 IFIP

A formal or computational specification of management policies is a prerequisite for the automation policy-based management. For that purpose, a common set of attributes has been proposed [1] and widely adopted to specify management policies. These are the followings:

- The *mode* or modality attribute specifies the obligation or authorization mode of policy. The obligation mode defines which activities a subject must (or must not) do, and represents a responsibility to achieve a given set of goals. The authorization mode defines which activities a subject is permitted (or forbidden) on a given set of target objects.
- The *subject* attribute specifies the objects to which the policies apply, i.e., those entitled to perform the policy activities.
- The *target* attribute represents the objects to which the policy is directed, i.e., the objects that are affected by policy's activities.
- The *action* attribute specifies the actions to be executed or those that are forbidden according to the authorization mode.
- The *constraint* attribute specifies the applicability of a policy.

Additional attributes, related to the importance of management goals and actions as well as to policies' behavior, can also be used. Examples include the policy priority [2], the policy life time [3], etc.

To express the previous policy attributes and thereby manipulate policies, we adopt the notation proposed in [1], where a policy is represented as follows:
Identifier Mode Subject {Action} Target [Constraint], Where:

- Identifier : identifies the policy.
- Mode = {A+ (for positive authorization policy), A- (for negative authorization policy), O+ (for positive obligation), O- (for negative obligation)}.
- Subject and Target scopes are specified using the so called scope expression as follows using set operations as follows:
  sc_expression :: = {objects} /* *set of objects* */
  sc_expression :: = sc_expression + sc-expression /* *set union* */
  sc_expression :: = sc_expression – sc_expresion /* *set difference*/*
  sc_expression :: = sc_expression ^ sc-Expresion /* *set intersection*/*
- Actions are specific to the managed objects. Typically for MIB variables actions are *read*, *write* and *notify* operations.
- Constraints specify the conditions that trigger or restrict the application of the policy.

Examples [5] of an authorization policy and an obligation policy are given bellow using the previous notation.

Example authorization policy:

Policy_1 A+ *Sregion_agents {enable(); disable(), reset()} *Sregion when (time (08: 00) && (18:00)

The policy identified by "Policy_1" authorizes the objects that belong to the subject domain Sregion_agents to perform enable(), disable() and reset() operations on the objects belonging to Sregion during the time period 08:00 and 18:00.

(c) 1999 IFIP

Example obligation policy:
Policy_2 O- *Sregion_agent2 {reset();off()} *Sregion

The policy identified by "policy_2" specifies that the subject into the *Sregion_agent2* must not perform the *reset()* and *off0* operations on objects belonging to the *Sregion* domain.

Our goal is to show how policy-based management can be supported within the standard SNMPv3 management framework. Although, the policy concept can be used for all SNMPv3-based management functionality, this paper emphasizes access control aspects. Therefore, only authorization policies are considered here to ensure that managing subjects are granted access to managed targets in an authorized and controlled manner. Prior to showing how authorization policies are enforced in SNMPv3 entities, we present, in the following section, an overview of the standard SNMPv3 framework particularly highlighting is access control model.

3. SNMPv3 Framework

The new version of the Internet-Standard Management Framework (referred to as SNMPv3) is derived from and builds upon both the original and the second Internet-Standard Management Frameworks (SNMPv1 and SNMPv2). All three Frameworks share the same basic structure and components.

Typically, an enterprise deploying the Internet-Standard Management Framework contains four basic components [10]:

- Several managed nodes, each endowed with an SNMP entity which provides remote access to management instrumentation (traditionally called an agent);
- At least one SNMP entity with management applications (traditionally called a management station or simply a manager);
- A management protocol (SNMP) used to convey management information between the previous entities;
- Management information structured in a standard way and stored into MIBs [7].

The SNMPv3 Framework builds on these four basic architectural components and uses the same layering principle to define new capabilities particularly in terms of security management. The new features of SNMPv3 include two main security concerns:

1. Data integrity, authentication and privacy;
2. Access control.

These are implemented by two distinct components, respectively the security subsystem and the access control subsystem, of the SNMPv3 management architecture described bellow.

The SNMPv3 specifications of the Internet-Standard Management Framework are based on a modular architecture [8]. The SNMP entity, either manager or agent, consists of an *SNMP engine* and one or several associated *applications*.

The SNMPv3 engine consists of the dispatcher, the message processing subsystem, the security subsystem, and the access control subsystem.

(c) 1999 IFIP

The *dispatcher* coordinates the communications between the various subsystems. Essentially, it determines to which application an incoming Protocol Data Unit (PDU) should be directed. The message processing subsystem is responsible for preparing outgoing messages and for extracting data from received messages. It may support several message processing models, for example SNMPv1, SNMPv3, etc.

The *security subsystem* provides message security services such as integrity, authentication and privacy. Multiple security models may be supported by the security subsystem. For instance, the User-based Security Model (USM), described in RFC2274 [9], is the standard security model currently used with SNMPv3. It provides integrity, authentication and privacy services by computing message authentication codes, key management and data encryption.

The *access control subsystem* constitutes a decision making point to allow or not a specific type of access (e.g., read, write, notify) to a particular object instance. Similarly, to the security services, the access control services can be provided according to multiple access control models to allow future updates in case the security requirements change. The View-Based Access Control Model (VACM), described in RFC2275 [10], is one such model currently used within SNMPv3 access control subsystem. It basically allows restricting access to a subset of the management information referred to as a MIB view.

At the application level of the SNMPv3 entity, the various applications ([11]) use the services provided by the SNMPv3 engine to accomplish specific tasks. According to the role of the SNMP entity (manager or agent), five dominant types of applications can be enumerated: command generators; command responders; notification generators; notification receivers; and proxy forwarders. The reader can refer to [8] for more details about these application types or other possible applications.

The subsystems, models, and applications within an SNMP entity may need to retain their own sets of configuration information. Portions of the configuration information may be accessible as managed objects. The collection of these sets of information is referred to as an entity's Local Configuration Datastore (LCD).

This paper emphasizes access control issues, which falls within the activity of the access control subsystem of the SNMPv3 engine. The VACM access control model is detailed in the following as this one is standardized within SNMPv3 management framework. The VACM model will be used in subsequent sections to enforce authorization policies.

The View-based Access Control Mechanism

The access control subsystem in the SNMP engine is responsible for checking if a specific type of access, such as read, write, notify, is allowed to a particular object instance. The decision to allow (or not) access is determined based on access control rules maintained by the SNMPv3 entity. The access control subsystem implements a single service isAccessAllowed() called by applications. Four tables are used during the verification process:

- vacmContextTable: Contains locally available contexts identified by contextName.

(c) 1999 IFIP

- vacmSecurityTogroupTable: This table maps a combination of securityModel and securityName into a groupName.
- vacmAccessTable: Contains access rights for groups.
- vacmViewTreeFamilyTable: Locally holds information about families of sub-trees within MIB views.

Input parameters to the service isAccessAllowed are the followings:

(a) securityModel: Security model in use;
(b) securityName: Principal who wants to access to object instance;
(c) securityLevel: Level of security;
(d) viewType: Read, write, or notify view;
(e) contextName: Context containing variableName;
(f) variableName: OID for the managed object which is made up of the object type object-type (m) and the object-instance (n).

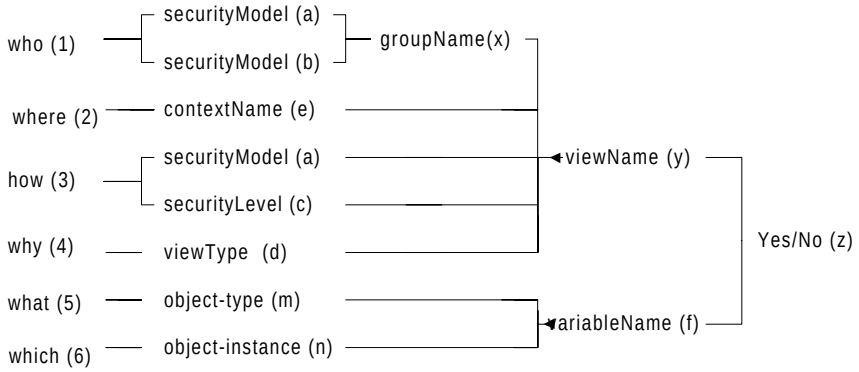

Figure 1: VACM mechanism

Figure 1 depicts the process of making the decision to grant access or not. This is done as follows:

- The partial "who" (1), represented by the securityModel (a) and the securityName (b), are used as the indices (a,b) into the vacmSecurityToGroupTable to find a single entry that produces a group, represented by groupName (x).
- The "where" (2), represented by the contextName (e), the "who", represented by the groupName (x) from the previous step, and the "how" (3), represented by securityModel (a) and securityLevel (c), are used as indices (e,x,a,c) into the vacmAccessTable to find a single entry that contains three MIB views.
- The "why" (4), represented by the viewType (d), is used to select the proper MIB view, represented by a viewName (y), from the vacmAccessEntry selected in the previous step. This viewName (y) is an index into the vacmViewTreeFamilyTable and selects the set of entries that define the variableNames, which are included in or excluded from the MIB view identified by the viewName (y).
- The "what" (5) type of management data and "which" (6) particular instance, represented by the variableName (f), is then checked to be in the MIB view or not, e.g., the yes/no decision (z).

(c) 1999 IFIP

Whenever an application calls the isAccessAllowed service of the access control subsystem, this one performs the following procedure based on the VACM model:

- The vacmContextTable is consulted to retrieve information about the SNMP context identified by contextName. If the desired information is not available in the table, then an errorIndication (noSuchContext) is returned to the caller.
- The vacmSecurityToGroupTable is consulted for mapping the securityModel and securityName to a groupName. If the information about this combination is absent from the table, then an errorIndication (noGroupName) is returned to the caller.
- The vacmAccessTable is consulted for information about the groupName, contextName, securityModel and securityLevel. Several cases are then possible:
  - If the requested information is not available, an error indication (noAccessEntry) is returned to the caller.
  - If the view to be used is the empty view (zero length viewName) or there is no view configured for the specified viewType, then an error indication (noSuchView) is returned to the caller.
  - If the specified variableName (object instance) is not in the MIB view, an error indication (notInView) is returned to the calling application.
  - Otherwise, the variable is in the MIB view, and access is allowed.

The local configuration datastore (LCD) contains configuration information local to the SNMP entity. Such information is modeled as managed objects and can be remotely configured. The VACM MIB is defined in the branch SMI MIB (1.3.6.1.6.3.16). As described previously, the VACM MIB is structured into four tables, namely vacmContextTable, vacmSecuritytogroupeTable, vacmAccessTable, and the vacmViewTreeFamilyTable. These tables implement access control rules. In order to allow for policy-based management applications, policy object attributes need to be into the corresponding entries in VACM tables. The following section describes such mapping.

4. Mapping of policy object attributes to VACM-MIB tables

The mapping of authorization policies into VACM tables is realized by a process, which translates policy templates into VACM MIB tables. The mapping process can be complex when policy templates are abstract and apply to a set of objects while Internet MIBs define scalar variables and tables.

It is worth mentioning that the policies processed by the implemented mapping process are the leaves of the enterprise policy hierarchy. In other words the considered policies are specific to the managed resources represented by the corresponding SNMPv3 agents. In this paper, we assume that the management platform includes policy distribution and refinement services. These services are provided by policy and domain servers that can be accessed by management applications to create, modify and remove management domains and policies. The mapping scheme of policy attributes to VACM tables is shown in Figure 2 and detailed in the following subsections.

(c) 1999 IFIP

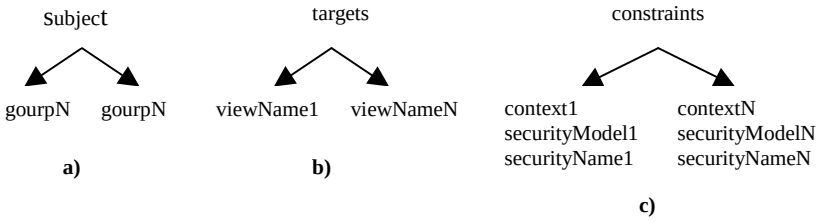

Figure 2: Mappings of attributes

4.1 Mapping policy subject

According to SNMPv3 specifications a group defines the access rights guaranteed to all security names that belong to this group. A security name represents a principal on behalf of, which services are provided, or processing takes place. The so-called principal can be an individual acting in a particular role; a number of individuals each acting in a certain role; an application, several applications; or any combination of these.

The subject attribute of an authorization policy specifies the objects to which the policy applies. It is given by a domain scope expression, which is specified in terms of explicit objects and domains. In case the used access control model is VACM, access rights are granted to a group. Hence, the subject represents one or several groups to which the policy applies. A different mapping can be required if a different access control model is used within the SNMP entity.

Mapping a subject attribute to groups is a one to many mapping. The mapping process generates the PDU actions that affect the SecurityToGroupTable. The mapping process adds the association between the securityName and the generated group if the latter doesn't exist in the securityToGroupTable. This is achieved as follows:

subject-groups (s: subject)

For all objects that define the subject s

Associate adequate groups;

If the group doesn't exist in securityToGroupTable

Then add the rows securityName-group in the securityToGroupTable;

EndFor;

A given authorization policy is applied to all users belonging to a group. This means that the VACM model assumes that an user must belong to a given group. Another limitation of the VACM model is the case where a person belongs to more than one group, which is often the case. Indeed, in such case, the person has to have several distinguished security names as in the SecurityToGroupTable a security name is unique.

4.2 Mapping policy target

The policy target attribute represents the objects to which the policy is directed. It specifies the objects on which the action(s), specified by the policy, can be performed. In the VACM, a group has access to a set of managed objects through the notion of

(c) 1999 IFIP

MIB view. A view specifies the objects accessible by the group. It is identified by a viewName, which is given in the vacmViewTreeFamilyTable.

Hence, the target attribute is specified by a set of viewNames. Consequently, the mapping of a policy to a viewName is a one to many mapping. The process of mapping policy target operates as follows.

Target-viewName (t:target)

For all objects that define the target t

Associate the viewName;

If viewName does not exist in the vacmViewTreeFamilyTable

Then create the view name entry in this table;

EndFor;

4.3 Mapping policy action

The action of an authorization policy specifies what the subject can do to the target. In practice, it corresponds to the name of a method of the target object. As mentioned above, the policies considered by the mapping process belong to the leaves of policies' hierarchy. This means that their actions should be conform to those supported by the SNMP entity, i.e., read, write and notify actions. Each action, specified in the actions' attribute, will affect the corresponding view in the vacmAccessTable.

According to the value of the read/write/notify attribute, the WriteViewName is, the ReadViewName or the object NotifyViewName affected. This way, the policy action attribute is used during the process of mapping policy target, but there is no direct mapping with the VACM tables.

4.4 Mapping policy constraints

The applicability of an authorization policy may be limited by specifying a policy constraint. The constraint may be applied to restrict the subject actions or limit the target space. A typical constraint is the one that places time restriction on the policy action such as an expiry date or an authorized time interval. We assume that this type of constraints is taken into account at the management application level.

In VACM, access is granted to a group if some conditions are verified: the level of security and the context in which the PDU is transported. For a member of the group different access rights can be defined for different security levels. In the User Security Model (USM), three security levels are defined, namely, noAuthNoPriv, authNoPriv and authPriv. In the first level, neither authorization processing nor privacy checking are performed when the SNMPv3 entity receives the PDU. In the second level, authorization processing without privacy checking is performed. In the third security level, both the authorization and the privacy processes are performed.

The context is a collection of management information accessible by an SNMP entity. An SNMP entity potentially has access to many contexts. A ContextName identifies a context.

The context, the security level and the security model are communicated by the PDU that requesting the access. In our implementation, policy constraints are limited to security level and to context name. As a result, the mapping of policy constraint a is

(c) 1999 IFIP

one-to-many mapping, as a manager can specify a sequence of constraints. The policy constraint mapping process is defined as follows:

constraints-securityLevel (c: constraints)

For each constraint sequence

Associate the security level;

Associate the contextName

EndFor;

Table1 summarizes the variables affected by the overall policy mapping process.

Table1: Mapping policy attributes to VACM MIB

Policy attribute	*VACM Table*	*SNMP Variables*
Subject	vacmSecurityToGroupeTable	vacmSecurityName vacmGroupName
Target	vacmAccessTable	vacmAccessReadViewName vacmAccessWriteViewName vacmAccessNotifyViewName
	vacmViewTreeFamilyTable	vacmViewTreeFamilyViewName vacmViewTreeFamilyMask vacmViewTreeFamilySubtree vacmViewTreeFamilyType
Constraints	vacmContextTable	contextName
	vacmAccessTable	vacmAccessSecurityModel vacmAccessSecurityLevel

The main policy mapping process used for enforcing management policies into the SNMPv3 framework is summarized by the following pseudo code:

PolicyMapping (subject, target, action, constraint)

For all policies

group= subject-group (subject);

If action = read then ReadViewName == Target-ViewName(target)

If action = write then WriteViewName == Target-ViewName(target)

If action = notify then NotifyViewName == Target-ViewName(target);

Security-levels == constraints-securityLevel(constraint);

add_to_VacmAccessTable (group, ReadViewName, WriteViewName, NotifyViewName, securitylevel)

EndFor

To illustrate the mapping process, let us consider the following example of VACM initialization parameters.

People that belong to the "initial" group are authorized to read, write, and notify the "Internet" View, if the context is null and if the USM Model is used with the "authPriv" security level.

This policy is expressed as follows:

Policy_1: (A+, "initial", "Internet", all accesses, authorization and privacy)

(c) 1999 IFIP

Examples of the resulting VACM tables generated by the mapping process are:

– vacmViewTreeFamilyTable:
  vacmViewTreeFamilyViewName : "internet"
  vacmViewTreeFamilySubtree : 1.3.6.1
  vacmViewTreeFamilyMask : ""
  vacmViewTreeFamilyType : 1 (included)
– vacmAccessTable:
  vacmGroupName : "initial"
  vacmAccessContextPrefix : ""
  vacmAccessSecurityModel : 3 (USM)
  vacmAccessSecurityLevel : authPriv
  vacmAccessContextMatch : exact
  vacmAccessReadViewName : "internet"
  vacmAccessWriteViewName : "internet"
  vacmAccessNotifyViewName : "internet"

5. Policy enforcement application: Implementation architecture

The policy mapping capability introduced in the previous section has been implemented as part of the ModularSNMPv3 project [12]. ModularSNMPv3 essentially implements an SNMPv3 framework as a set of configurable modules written in Java. It allows dynamic binding and unbinding of the modules at run time as well as their remote configuration. The overall Modular SNMPv3 system is operational and available on the Web at [14]. In addition a set of reusable classes of managed objects are created using the large number of standardized managed objects. These classes together with the reusable SNMPv3 framework modules are made available to developers of secure network management applications.

The overall implementation architecture is depicted by Figure 3. As mentioned previously, our implementation consists of a set of modules with their respective MIBs. These modules are structured into an engine layer and an application layer. The implemented engine includes the following modules:

- The dispatcher, which coordinates the communications between the modules in the engine and the application layer. Basically it direct incoming SNMP PDUs to the appropriate application.
- The SNMPv1 and SNMPv3 message processing modules, responsible for preparing outgoing messages and extracting data from incoming messages. The SNMPv1 module implements SNMPv1 messages format while SNMPv3 module process SNMPv3 messages format. These modules allow SNMPv3 secure management as well as interoperability with integrated exiting SNMPv1 entities.
- A security subsystem is implemented based on the User Security Model (USM) specified in RFC2274 [9]. It is composed of three modules (SHAModule, DESModule and MD5Module) and the corresponding MIBs (USM User MIB and Stats MIB). The modules support the security features of data integrity, authentication and privacy.
- Finally an access control module is implemented based on the VACM model (dark Grey box at the top of Figure 3) supported within the so-called VACM MIB.

(c) 1999 IFIP

At the application layer of the SNMPv3 entity, a number of specific purpose applications are implemented including the proxy, the requester, and a number of MIBs and MIB manipulation modules. Details of these applications can be found in [11]. The implemented application of interest here is the policy enforcement module incorporated at the application layer of the SNMPv3 entity as shown in Figure 3 in a dark Grey box.

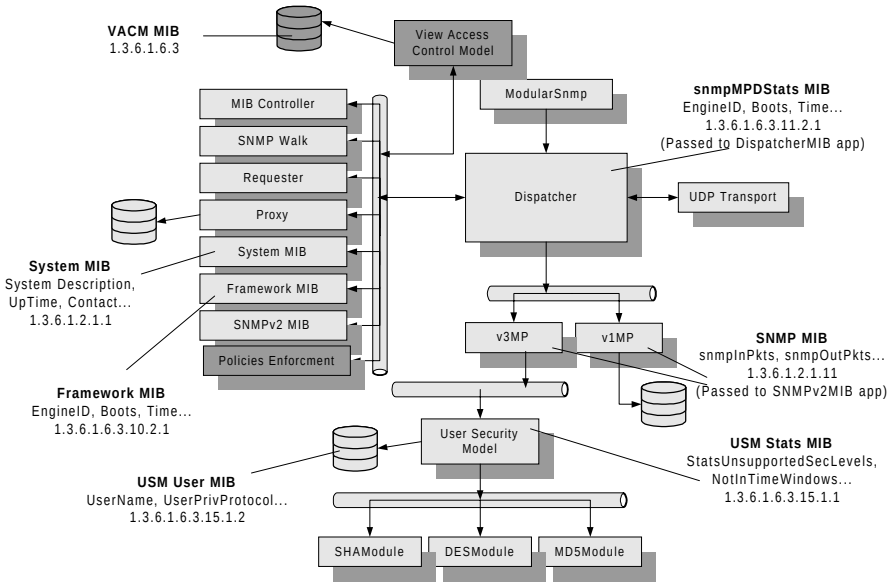

Figure 3: ModularSNMPv3 architecture

The mapping process is implemented by an application incorporated at the application layer of the SNMPv3 entity that represents an SNMPv3 agent such as the one depicted by Figure 3. Authorization policies are modeled as objects and stored in the Local Configuration Datastore (LCD) within the SNMPv3 agent. They can be modified remotely by an authorized SNMPv3 manager entity or an authoritative user identified by a security name. The mapping process runs in the agent entity and is launched whenever access to managed objects represented by the agent is requested. Its role is to map the authorization policy attributes to the appropriate access control model supported by the agent, i.e., VACM in our implementation. Requests of the manager, which are sequences of actions, are encapsulated by appropriate SNMP PDUs. When receiving these SNMP PDUs, the agent enforces the encapsulated policies by mapping them to the corresponding VACM MIB tables. As a result, SNMP-SET actions, mainly, change MIB attributes' values. The mapping rules between authorization policies and view based access control implemented by the mapping process are those described in section 4. The advantage of our implementation scheme is that the administrator and management applications only manipulate policies and enforce them in the managed system. The security model implemented by SNMPv3 entities is transparent to the management application and can be changed without affecting this one. Indeed, the standard SNMPv3 management

(c) 1999 IFIP

framework is designed to support multiple and different security models. Policy-based management applications will manipulate object-oriented policies such as those described in this paper regardless of the security model supported by the managed system or the SNMPv3 agents representing the managed system resources. Note that the same access control scheme applies when remote managers access VACM tables in the LCD to (re-)configure them.

6. Conclusion

The recently standardized version of the SNMP management protocol, SNMP version 3, has filled the gap left by the earlier versions of the protocol in terms of security. This lack of security has, until recently, hampered the acceptance and deployment of the SNMP protocol in security-sensitive enterprise network management. An enterprise has its own management policies that reflect its corporate and management goals. Such policies have to be taken into account in the process of managing the network resources employed by the enterprise in order to achieve the goals of this enterprise. These last years research and developments have witnessed the maturity of the policy-based management approach. Indeed, informational and computational models for expressing and manipulating policies are now available.

The work presented in this paper was devoted to the implementation of policy-based management in the standard SNMPv3 management framework. It particularly focused on security policies and their enforcement into SNMPv3 management entities. In the SNMPv3 framework, security is handled by the security subsystem and the access control subsystem of the SNMPv3 entity (manager and/or agent). In the first subsystem, security rules are used for message encryption, authentication, and privacy, while in the second one they are used for checking access rights. The approach of policy enforcement in the SNMPv3 framework presented in this paper has been illustrated by emphasizing authorization policies and their mapping to access control attributes maintained by the access control subsystem. We particularly considered the View-based Access Control Model (VACM) standardized within the SNMPv3 framework and implemented a mapping of authorization policies to VACM MIB tables. The mapping process is provided by a gateway application incorporated at the application layer of the SNMPv3 entity. It is worth mentioning that an equivalent mapping process can be performed for obligation policies, for example to demand the implementation of an authentication process according to a given authentication protocol. In this case, the example obligation policy will be mapped to a security subsystem MIB variable such as the *UsmUserAuthProtocol*, specified by the standard User Security Model (USM).

By introducing policies into SNMPv3, we allow the administrator or the management application developer to manipulate policy objects rather than SNMP access control variables. Furthermore, in our implementation policy objects are enforced into the SNMP agent are maintained in the entity's Local Configuration Datastore (LCD) which can be accessed and modified remotely, thanks to the configurability feature of the implemented SNMPv3 framework. This allows customizing the access control and security models (VACM and USM) implemented by the entities or use alternative models according to the enterprise-specific security

(c) 1999 IFIP

requirements. In all cases the security models supported by the SNMPv3 entities are transparent to management applications.

The limitation of our framework, which is shared by the majority of gateway-based approaches, is the performance issue. Indeed, the mapping process introduces an additional processing cost and delay. However, this is the price to pay to allow for open but secure policy-based management.

As to the future, we plan to extend our gateway application to integrate other security models. Also a demonstrator is being built to support differentiated services IP over an ATM network endowed with SNMPv3 capable agents. The demonstrator also includes a policy server being developed according to the Common Open Policy Service and protocol [13] to be used by policy-based network management applications.

References

[1] Sloman, M., "Policy Driven Management for Distributed Systems", in Journal of Network and Systems Management, vol.2 part 4, 1994.

[2] Boutaba, R., "A Methodology for Structuring Management of Networked Systems", in IFIP Transactions, pp. 225-242, North-Holland, 1994.

[3] Wies, R., "Policies in Integrated Network and Systems Management", Phd Thesis, June 1995.

[4] Alpers, B., Plansky, H., "Concepts and Application of Policy-based Management", Proceeding of the 4 International Symposium On Integrated Network Management, Santa Barbara, IFIP, Chapman and Hall, May 1995.

[5] Lupu, E., Sloman, M., “Conflict Analysis For Management Policies” Proceeding of the 5 International Symposium On Integrated Network Management IM’97, Santa Barbara, IFIP, Chapman and Hall, May 1997.

[6] Koch, T., Kramer, B., Rohde, G ., "On a Rule Based Management Architecture", Proceeding IEEE 2 International Worshop on Services in Distibuted and Networked Environments, Whistler, BC, Canada, June 1995.

[7] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "The Simple Network Management Protocol", STD 15, RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance International, and the MIT Laboratory for Computer Science, May 1990.

[8] Harrington, D., Presuhn, R., B. Wijnen, "An Architecture for describing SNMP Management Frameworks", RFC 2271, January 1998.

[9] Blumenthal,U., Wijnen,B., “User-based Security Model (USM) for version 3 of Simple Network Management Protocol (SNMPv3)”, RFC 2274, November 98

[10] Wijnen, B., Presuhn, R., K. McCloghrie, "View-based Access Control Model for the Simple Network Management Protocol (SNMP)", RFC 2275, January 98

[11] Levi, D., Meyer, P., B. Stewart, "SNMPv3 Applications", RFC 2273, January 98.

[12] Cherkaoui, O., Saint Hillaire, Y., Mili,H., Serhouchni,A., "Towards a modular and interoperable SNMPv3", IEEE NOMS’98, New Orleans 98.

[13] Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan , R., Sastry, A., "The COPS (Common Open Policy Service) Protocol ", Internet Draft. November 98.

[14] Working Java-based Modular SNMPv3 http://www.teleinfo.uqam.ca/snmp.

(c) 1999 IFIP