Role-Based Access Control for XML Enabled Management Gateways

V. Cridlig, O. Festor and R. State

LORIA - INRIA Lorraine
615, rue du jardin botanique
54602 Villers-les-Nancy, France
Email: {cridligv,festor,state}@loria.fr

Abstract. While security is often supported in standard management frameworks, it has been insufficiently approached in most deployment and research initiatives. In this paper we address the provisioning of a security "continuum" for management frameworks based on XML/SNMP gateways. We provide an in depth security extension of such a gateway using the Role Based Access Control paradigm and show how to integrate our approach within a broader XML-based management framework.
Keywords: management gateways, SNMP, XML-based management, security, key management.

1 Introduction

Security needs in network management appeared when administrators realized that malicious parties can use this vector to either get confidential information like configuration and accounting data or even attack the network and the devices through management operations. Integrity, authentication, privacy, replay and access control are of major importance to the distributed management plane. Network management data is sensitive and can compromise the security of the whole network. For example, firewall tables, routing tables, routers' configuration can help attackers to discover network security holes and the network topology. This is the reason why extending management protocols with security features is crucial.

Security in distributed systems is usually built around five areas: authentication, integrity, confidentiality, availability and access control. Authentication process gives warranties that a remote principal is the one he claims to be. Integrity process assures that the transmitted data has not been altered during transmission. Confidentiality includes encryption and decryption processes to ensure that data cannot be read by a third party. Availability ensures that a service is accessible over time. Access control, which can not be performed without an authentication phase, allows to restrict access on data to some authorized principals.

These issues are even more important in a utility computing type of environment. Dynamic, on-demand usage of resources provided by a third party over a well delimited time-interval requires a flexible management plane. Flexibility

in this case is related to the degree of configuration accessible to a resource consumer while these resources are supporting his business.

The problem addressed in this paper is twofold: firstly, we consider how to maintain the overall security level, when migrating from SNMP-based to XML-based management. Such a migration can be assured with an XML/SNMP gateway (see [1] and [2]). The XML/SNMP gateways address the interoperability among XML managers and SNMP agents and should not introduce security holes but rather should establish a security continuum. In particular, a non-authorized principal should not be able to manage agents either directly or through the gateway. These requirements suppose a mapping and a policy coherence maintenance for the different security models involved in the whole architecture. The second issue addressed by our paper concerns the security provisioning of a management gateway required in dynamic utility computing environments, where on demand resources must allow a temporary management session by a foreign manager. A manual per device configuration is not scalable if the utilities are numerous. A more realistic solution is to use a management gateway, which mediates the access to the managed devices.

The remainder of this paper is organized as follows. In section 2, we summarize the existing XML/SNMP gateways as well as the SNMPv3 functional architecture and its security modules, namely User Security Model (USM) and View-based Access Control Model (VACM). In section 3, we propose an XML/SNMP gateway extended with the desired security features. Section 4 concludes this paper and opens perspectives to our work.

2 State of the art

2.1 Existing XML/SNMP gateways

Over the past few years, the eXtensible Markup Language (XML [3]) has proven to be an excellent way to build interoperable applications. Many XML standards and recommendations emerged mainly from the World Wide Web Consortium (W3C [4]) in order to avoid proprietary solutions and then improve interoperability. The advantages of XML-based techniques such as XML document transmission and processing as well as embedded security are also recognized in the management plane [2]. However, the large number of existing SNMP agents slows down the XML-based management solutions deployment. Therefore, intermediate solutions using XML/SNMP gateways are required to provide a temporary solution before a potential full and complete migration to XML-based management solution.

XML/SNMP gateways allow managers to communicate with their agents using XML technologies even if the agents are solely SNMP compliant. The managers can then benefit of XML tools to process and/or display XML documents conveniently, or to express complex relationships among managed services as in [5]. The gateway is responsible for translating XML requests coming from the managers to SNMP requests towards the agents. The mechanism is reversed for

the agents responses. Mapping XML to SNMP means translating both exchange protocols and information models.

Yoon, Ju and Hong [6] proposed a full SNMP MIB (Management Information Base) to XML Schema translation algorithm. The latter is based on the following document structure conversion. A SNMP SMI (Structure of Management Information) node becomes an XML Schema element, a node name becomes an element name and clauses in node become attributes of the element. One important topic (feature, component, ...) of the proposed approach is the SMI data types translation. XML Schema provides an elegant way to define equivalent data types using default data types and patterns. Note that all XML nodes have an Object Identifier (OID) attribute corresponding to the MIB node location in order to facilitate data retrieval. For a given MIB, this translator produces both the XML Document Object Model (DOM) tree structure which is used to store management data in the gateway and also to create XML documents and the XML Schema file for validation.

Within the context of the gateway implementation, Yoon, Ju and Hong proposed three approaches for the manager/gateway communication: a DOM-based translation, an HTTP (Hypertext Transfer Protocol)-based translation and a SOAP-based translation. These approaches, their advantages and drawbacks are discussed in [1].

F. Strauss and T. Klie [2] also proposed a SMI MIB to XML Schema definitions converter. Although this converter is also based on a model-level approach (see [7]), their approach is quite different. Instead of using OIDs to process the mapping, the latter is based on namespaces: each MIB module is translated into an XML Schema and identified with a namespace to uniquely map data elements. This mapping is driven by the intention to produce an XML document close to the XML philosophy (meaning both human and machine readable). Both approaches (i.e. [6] and [2]) are elegant. A translator (mibdump) analyses a whole MIB, creates the XML Schema, generates the associated set of SNMP requests, collects the data from the agent and builds the valid (regarding the XML Schema) XML document. However, mibdump can only collect data from a single MIB.

We are working on the development of an enhanced gateway based on a role based access control paradigm. This paper describes some of its features. While being conceptually based on HTTP and XPath for manager to gateway communications and DOM for data storage, we extend this architecture with security modules to allow authentication, privacy and access control.

2.2 SNMPv3

The SNMPv3 (Simple Network Management Protocol [8, 9, 10]) functional architecture was designed to address potential security issues. To achieve this security challenge, this architecture integrates two subsystems. The first one, called security subsystem, consists in a privacy module, an authentication module which provides both authentication and integrity and a timeliness module. It is implemented with the User-based Security Model (USM [11]) by default. The second

one, implemented with the View-based Access Control Model (VACM [12]) is an access control subsystem. These integrated security tools allow SNMP managers to perform secure requests on SNMP agents.

USM authentication and confidentiality services are based on a shared secret localized key paradigm. Each manager/agent pair uses a particular key pair (one key per service). The security level required by the manager dictates the message preparation or checking depending on whether the message is incoming or outgoing.

VACM performs access control based on different mib tables storing access policies. Authorization decision is based on input parameters like the object OID, the operation to be performed, the requesting security name, the security model and level. As every object of the MIB tree, VACM policies can be configured remotely provided that the requester has sufficient rights to modify these tables.

3 The proposed security framework

3.1 Motivation

The existing XML/SNMP gateways focused mainly on the operational part of network management while security issues were not yet addressed. The application of the XML/SNMP gateway to a real network management environment requires the incorporation of additional security mechanisms, since its deployment without efficient security features provides major security breaches. The major motivation for our work is to propose a coherent security framework for SNMP/XML gateways, realizing a "security continuum", i.e. assuring that security policies are independent of the underlying used management protocol (native SNMP, SNMP/XML gateways, or native XML). We present here a security extension for the XML/SNMP gateways addressing the authentication, confidentiality and authorization issues. These security services intend to provide the ability to perform more sensitive configuration operation, which is one of the shortcomings of SNMP [2]. Such a "security continuum" is essential if a management gateway is used in a utility computing scenario. The sound configuration of many devices at once in order to temporarily serve a third party manager requires on the one hand to assure that only allowed operations are possible, and on the second hand that the security configuration be fast. We propose an access control enabled XML/SNMP gateway which meets these requirements.

3.2 Requirements

The gateway provides a unified security policy based on Role Based Access Control (RBAC [13]). The access control module must be able to create sessions for users and map the access rights of each user on the SNMP agents on the fly. This makes it possible to manage a unique centralized policy and deploy it in a transparent way for users. Moreover authoring authorization policies with RBAC is proven to be easy, scalable and less error-prone. A user creates an RBAC policy on the gateway. Since this policy is mapped on the SNMP agents VACM tables, no change is required within SNMPv3 architecture.

3.3 Functional architecture

Our secure XML/SNMP gateway, depicted in figure 1 embeds an SSL layer to allow secure communications between the managers and the gateway. This layer is necessary to allow manager identification which is, in turn, necessary to perform access control process. First, the manager initiates an encrypted SSL session on the gateway. Then the manager is identified by filling an HTML form with his login/password credentials.

Moreover, SNMPv3 new security features are difficult to use and suffer from lack of scalability [14]. Therefore, our architecture uses an authorization model (RBAC) different and therefore independent from the SNMPv3 one. RBAC allows high level and scalable access control process and configuration. The RBAC model consists in a set of users, roles, permissions (operations on resources) and sessions. The originality of RBAC model is that permissions are not granted to users but to roles, thus allowing an easy reconfiguration when a user changes his activity. A role describes a job function within an organization. The permission to role relationships illustrates that a role can perform a set of operations on objects (privileges). In the same way, the user to role relationships describes the available function a user is allowed to endorse in the organization. Lastly, a session gathers for each user his set of currently activated role, on which behalf he can interact with the system.

In this paper, we consider the NIST (National Institute of Standards and Technologies) RBAC (Role-Based Access Control [13]) model which gathers the most commonly admitted ideas and experience of previous RBAC models. This standard provides a full description of all the features that should implement an RBAC system. RBAC model allows the description of complex authorization policies while reducing errors and costs of administration. Introduction of administrative roles and role hierarchy made it possible to reduce considerably the amount of associations representing permissions to users allocation.

Consequently, our architecture embeds an *RBAC manager* for authorization decision process consisting in an *RBAC system* and an *RBAC repository*. In order to still allow pure SNMPv3 configuration (i.e. direct manager/agent management for SNMPv3-based managers), it must be possible to push authorization policies onto the SNMP agents. Since RBAC and VACM models are different, a mapping module, *RBACToVACM translator*, is also necessary.

The *RBAC repository* owns an authorization policy. This policy describes:

- the set of users who are allowed to interact with the system,
- the set of roles that can potentially be endorsed by users,
- the set of scopes (objects) on which permissions are granted,
- the set of permissions which consist in an operation an a scope,
- the set of UAs (user assignment) which describe user to role association and the set of PAs (permission assignment).

An example of such a policy is depicted in figure 2.c.

The *RBAC system* implements all the features needed to handle authorization data and to process access evaluation. It is possible:

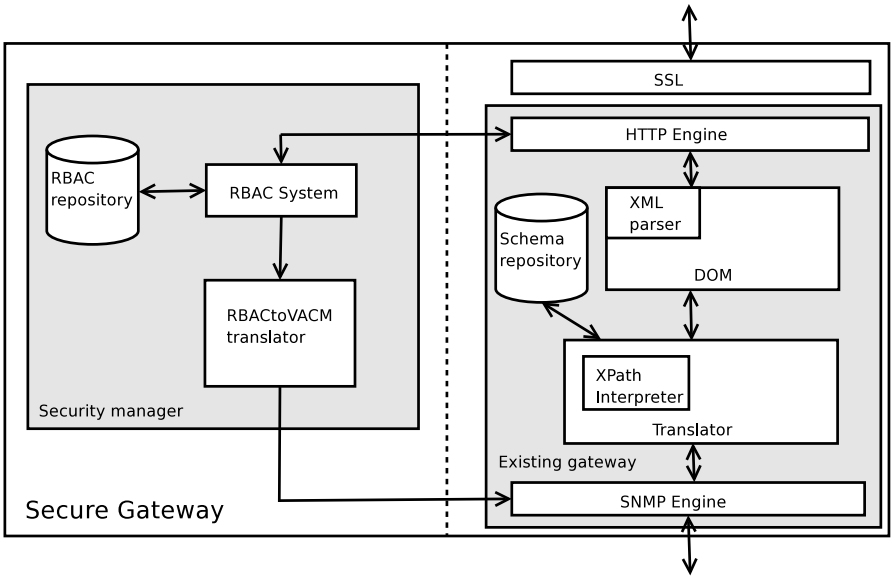

Fig. 1. Secure gateway functional architecture

- to create, modify or delete the different RBAC elements (users, roles, permissions) in order to build policies according to our needs,
- to manage (create and close) user sessions,
- to add or remove active roles from a session,
- to evaluate the access requests regarding the policy.

Moreover when a new role becomes active, it calls the *RBACtoVACM translator* in order to update VACM policies on agents. Most of the features described here are detailed in [13].

The *RBACToVACM translator* is responsible for mapping the gateway RBAC policy on the agent VACM policy. Different XML documents are needed to perform this mapping. We describe them in the mapping section.

3.4 Authorization

Linking RBAC to management data The XML document depicted in figure 2.c describes an RBAC policy scenario. Different existing languages such as XACML [15] Profile for RBAC or .NET MyServices Authorization language [16] can model an RBAC policy. Although the first language is acknowledged to be compliant with the NIST RBAC standard [13] and the second one is quite elegant, we propose our own language which is as close as possible to the NIST RBAC model and terminology while remaining as simple as possible for the purpose of an XML/SNMP gateway prototype. The XML representation of the RBAC model is of minor importance. We propose here to map a model, not a particular XML representation of the model.

Each RBAC element is described independently and owns an identifier (Id) attribute such that the user to role (UA) and permission to role (PA) assignments can reference them. Users (here bob and alice) consist in a login and a password. Roles are described with names. Scopes reference the XML management data of the figure 2.a using XPath. XPath is quite simple and allows to address powerfully a set of elements in an XML document. For instance, scope s1 designates all the *ifEntry* elements of the XML management data. This relationships is shown with the first arrow. Each permission has a reference to a scope and an operation. Only a subset of XPath possible expressions is allowed in our access control policies since the *vacmViewTreeFamilyTable* can only contain OIDs. For instance, an XPath expression referencing a particular attribute value can not be mapped to VACM.

Note that the XML resources depicted in figure 2.a are conforming to the XML Schema generated from the IF-MIB and partially depicted on figure 2.b. For instance, *ifPhysAddress* structure is described (see the second arrow) in the XML Schema. It is important to note that each element of the schema contains the corresponding OID for data retrieval. This will be used for the mapping of our RBAC policy on VACM tables as illustrated with the third arrow.

Mapping RBAC/VACM (one way mapping) In order to map dynamically the RBAC policy on SNMP agents, we have to translate XML RBAC policy into SNMP VACM tables. We provide, in this section an algorithm to map RBAC users on USM users, RBAC roles on VACM groups, scopes on views, and permissions to *vacmAccessTable* entries. The mapped access control has to implement the initial policy behavior. Although sessions can not be expressed in VACM, it can be simulated by mapping only the activated roles for a given user. The VACM tables reflect the activated roles, permissions and users of the RBAC model. The RBAC model is not fully mapped on the agent VACM tables, i.e. permissions are loaded on agents on demand.

The different steps of the mapping algorithm are the following: first, we create a group in the *vacmSecurityToGroupTable* for each user u; then, we add an entry in the *vacmAccessTable* associated to three views. These views are built in the *vacmViewTreeFamilyTable* and gather all objects allowed sorted by "read", "write" or "notify" access type.

Let us consider the example developed in figure 2.c. Our approach consists in collecting all permissions associated to all active roles of a given user. A user-specific group gathering all his active permissions can be created. The fourth arrow of figure 2.d illustrates that a new entry in the *vacmSecurityToGroupTable* is created for each RBAC user (bob for instance) with USM as default security model and a group reflecting the user SecuritName (bobGroup). Figure 3 shows the pseudo-code algorithm of a role activation immediately after the login step.

Permissions are sorted by operation, thus building available views for that user. The algorithm uses the XML Schema from the repository (figure 2.b) in order to retrieve OIDs corresponding to a scope. Note that a scope contains an XPath and *mibns* attribute bound to a particular MIB. The *vacmAccessTable*

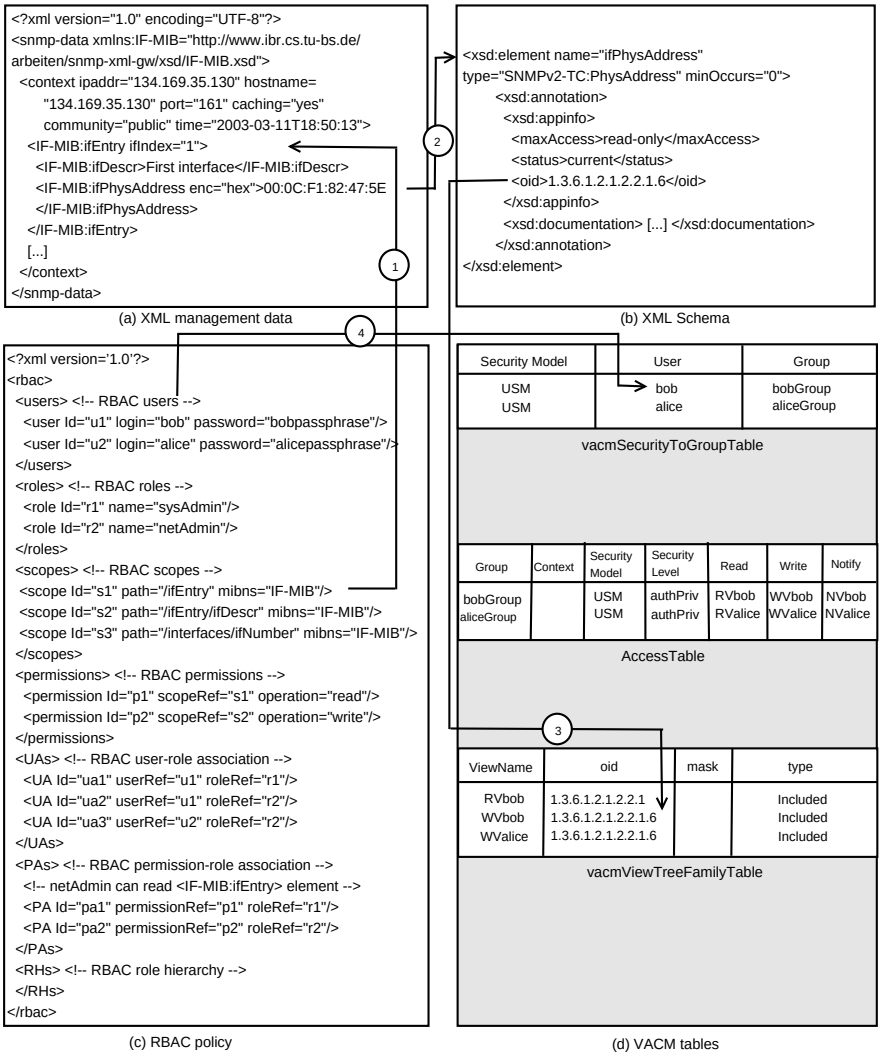

Fig. 2. RBAC/VACM mapping

depicted in figure 2.d will contain one entry for each group. For instance, in order to build the bob's read view, we have to gather all bob's active permissions whose operation is *read*, retrieve the OID using the XML schema and then add the corresponding entries in the *vacmViewTreeFamilyTable* to build a sophisticated view.

BobGroup is the union of both SysAdmin and NetAdmin roles. Roles may be associated to several shared permissions. Without separation of duty constraints, Bob can activate its two available roles at the same time in a session.

```
addActiveRole(user u, role r){
  // Example: ("USM", "Bob", "Bobgroup")
  AddSecurityToGroupTableEntry(USM, u.securityName,
                               u.securityName+"group");

  // Example: (Bob, "", USM, authpriv, "", "RVBob", "WVBob", "NVBob")
  AddVacmAccessEntry(u.securityName, "", USM, authpriv, "",
                     "RV"+u.securityName, "WV"+u.securityName,
                     "NV"+u.securityName);

  Foreach permission p of r{
    // Example: ("R"+"V"+"Bob","1.3.6.1.2.1.2.2.1.6","FF","included");
    addVacmViewTreeFamilyEntry(p.operation+"V"+u.securityName,
                               p.getOid, mask, included);
  }
}
```

Fig. 3. RBAC to VACM mapping algorithm

Consequently, Bob can read 1.3.6.1.2.1.2.2.1 and write in 1.3.6.1.2.1.2.2.1.6 subtree. The result *vacmViewTreeFamilyTable* will have the entries shown in figure 2.d. Permissions are updated dynamically on the SNMP agents only when a user requests to add or drop an active role. It is still possible to add separation of duty constraints which are controlled on the gateway side: since the gateway is the only entity which can set permissions on SNMP agents, the gateway can deter a user from adding an active role depending on policy constraints.

This is the basic algorithm. There may be several identical entries in the *vacmViewTreeFamilyTable*. We optimize also on the number of entries in the *vacmViewTreeFamilyTable* when read access is allowed to both a tree and one of its subtrees in the same view. Note that we use only USM associated with *AuthPriv* security level because we chose the RBAC NIST standard which does not describe contexts (do not confuse RBAC context and SNMP context which is totally different). However it is possible to add new views for different security levels. This RBAC context should be first described using the RBAC model as described in [17]. This mapping algorithm serves as a base to introduce RBAC module in the XML/SNMP gateway.

In order to improve the ease of use of the RBAC manager, it is very promising to use XPath object addressing. XML DOM can then be used to translate XPath expression into OIDs and VACM views. It also makes it possible to perform XML-level access control.

Access control process Each manager (user in RBAC terminology) owns a login / password. This login / password is a shared secret between the manager and the gateway. These credentials (in fact the password) are used to generate the SNMP keys for authentication and privacy. Consequently, both the manager and

the gateway are able to perform security requests since they know the manager's password. This way, a manager can manage SNMP agents without going through the gateway provided that permissions are deployed on the managed agent.

Managers have a permanent RBAC session. A manager must explicitly close an RBAC session. When this happens, all managed agents for that manager should be reconfigured, i.e. all permissions for that manager should be deleted. A manager can log out without closing its RBAC session, because his rights are still valid on agents. Managers can activate and deactivate roles when needed. If a manager logs out of the gateway without closing its RBAC session, the gateway does not remove the access rights on the agent so that the manager can still perform SNMPv3 requests on the agent without using the gateway. In our approach, RBAC sessions are persistent. This avoids mapping RBAC on VACM or removing VACM access rights each time a manager logs on or logs out of the gateway. Hence a decreasing number of VACM requests.

When a manager activates a role on the gateway (for an agent), the gateway updates permissions available on the agent VACM tables. In order to update permissions on an agent, the RBAC system maps the RBAC permissions of all active roles of this user.

When the gateway receives a request from a manager, it uses the login / password of this manager to generate the SNMP request so that the agent can know the user performing the request and then control access. This is transparent to the agent. The login / password of each manager is stored as part of the RBAC model in the "user" object. To simplify the login / password management, the gateway uses the same login / password for manager identification on the gateway and on the agents with SNMPv3. However, this is not a strong constraint. This way, managers must remember a single login / password pair.

When a manager wants to access a new device through the gateway, the gateway creates an account for him (provided the user is in the RBAC model of that group) on the agent by cloning itself. The user activates a role on the gateway. The gateway maps the associated permissions on VACM tables of the agent. Then, the user can start performing operations on the agent either through the gateway using HTTP or directly on the agent using SNMPv3. In this way, newly arrived devices are auto-configured for access control. The only requirement is that the gateway must know a valid user login / password as a starting point on each agent.

A special user should have particular rights to modify the RBAC policies on the gateway (i.e. performing maintenance RBAC operations, like createPermission(), addUser(), assignUser() with the NIST RBAC standard model). This user defines roles, users and permissions in a high level view. Mappings from this RBAC level to the VACM level is transparent to the special user. Other users can only open and close RBAC sessions, add active roles, drop active roles.

3.5 Authentication and confidentiality

Each manager has his own account within the gateway so that he can log on using his login/password credentials. In order to avoid sending these credentials

in the clear, the gateway and a manager first establish a secure session using security mechanisms such as Secure Socket Layer (SSL [18]).

The manager password is also used to generate the SNMPv3 secret keys. Since both the manager and the gateway know this password, both can generate the keys needed to authenticate and encrypt the SNMPv3 messages. This way, both can send SNMPv3 requests to a given agent. The advantage is that a manager can still request an agent even if the gateway is down.

A bootstrap phase works as following: the gateway has a default account on all SNMPv3 agents. When a manager needs to request an agent, the gateway clones itself on the agent, thus creating a new SNMP account for this manager. In net-snmp, a new account does not inherit the access rights of the account from which it is cloned since no entry is added in the VACM tables: only the credentials are duplicated. Therefore, there is no risk that a new user can have the same permissions as the gateway. The gateway changes the remote password of the manager according to the manager's password in the local RBAC system. Then the gateway grants access rights to the manager depending on the RBAC policy and the active roles.

4 Conclusion and future work

The problem addressed in this paper is the lack of security introduced by the use of XML/SNMP gateways. These gateways provide powerful means to manage device and therefore should be protected with adapted security mechanisms. We proposed some security extensions to an XML/SNMP gateway in order to provide a high level of security and simplify the configuration task of SNMPv3 security managers.

The benefits of our approach are the following:

- security continuum for XML/SNMP gateways,
- uniform scalable access control policies for network management elements using the RBAC model,
- on-demand access rights distribution from the gateway to the SNMP agents using a RBAC to VACM mapping algorithm. Only needed access rights (not all the RBAC policies) are mapped on agents VACM policies,
- easy automated VACM configuration.

An early prototype has been implemented within our group. It extends the servlet based SNMP/XML gateway (SXG) implemented by Jens Mueller. We defined a simple XML Schema (http://www.loria.fr/∼cridligv/download/rbac.xsd) modeling RBAC, which is also used to automatically generate Java classes. The prototype is made of two separate parts, both secured with SSL and accessible after a login/password phase: while the first part is an web RBAC editor used by a super user to setup authorization configuration, the second part is the XML/SNMP gateway itself allowing management operation after an explicit role activation request from the manager.

In the future, there is a great interest in applying an RBAC policy on the gateway to a group of devices. Maintaining one policy for each device is not scalable. Maintaining one policy for all devices is not flexible enough because some devices are more sensitive than others and may implement different MIBs. This justifies our approach to attach an RBAC model to each group of devices. When an RBAC model changes, all devices belonging to that group are updated at the access control level (i.e. VACM tables). Although we define several RBAC models, the set of users remains the same for all of them. This way, a user managing several groups of devices owns a single password. However, roles can be different in two different models and a role can be bound to several RBAC models.

When using the gateway, we can also consider that the access control is made inside the gateway thus avoiding a request which would not be allowed. However the current permissions are still mapped on agent to keep the RBAC model state coherent with the VACM tables.

In a mobility context, we can imagine that different network domains host different XML/SNMP gateways. When a device moves from one domain to another one, the local gateway should have access rights to configure the visitor device. Inter-gateways negotiations could be envisaged to allow the local gateway to create SNMP accounts for the local managers to perform minimal operations.

References

1. Oh, Y.J., Ju, H.T., Choi, M.J., Hong, J.W.K.: Interaction Translation Methods for XML/SNMP Gateway. In Feridun, M., Kropf, P.G., Babin, G., eds.: Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002. Volume 2506 of Lecture Notes in Computer Science., Springer (2002) 54–65
2. Strauss, F., Klie, T.: Towards XML Oriented Internet Management. In Goldszmidt, G.S., Schönwälder, J., eds.: Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM 2003). Volume 246 of IFIP Conference Proceedings., Kluwer (2003) 505–518
3. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation (2004)
4. W3C: World Wide Web Consortium (W3C). (http://www.w3.org)
5. Keller, A., Kar, G.: Determining Service Dependencies in Distributed Systems. In: Proceedings of the IEEE International Conference on Communications (ICC 2001), IEEE (2001)
6. Yoon, J.H., Ju, H.T., Hong, J.W.: Development of SNMP-XML Translator and Gateway for XML-based Integrated Network Management. International Journal of Network Management **13** (2003) 259–276
7. Martin-Flatin, J.P.: Web-Based Management of IP Networks and Systems. Wiley (2003)
8. Case, J., , Mundy, R., Partain, D., Stewart, B.: Introduction and Applicability Statements for Internet Standard Management Framework. STD 62, http://www.ietf.org/rfc/rfc3410.txt (2002)
9. Stallings, W.: Network Security Essentials. Prentice Hall (2nd edition 2002)

10. Subramanian, M.: Network Management, Principle and Practice. Addison Wesley (1999)
11. Blumenthal, U., Wijnen, B.: User-based Security Model for version 3 of the Simple Network Management Protocol (SNMPv3). STD 62, http://www.ietf.org/rfc/rfc3414.txt (2002)
12. Blumenthal, U., Wijnen, B.: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). STD 62, http://www.ietf.org/rfc/rfc3415.txt (2002)
13. Kuhn, R.: Role Based Access Control. NIST Standard Draft (2003)
14. Lee, H., Noh, B.: Design and Analysis of Role-Based Security Model in SNMPv3 for Policy-Based Security Management. In: Proceedings of the International Conference on Wireless Communications Technologies and Network Applications, ICOIN 2002. Volume 2344 of Lecture Notes in Computer Science., Springer (2002) 430–441
15. Anderson, A.: XACML Profile for Role Based Access Control (RBAC). OASIS Committee Draft (2004)
16. Microsoft: Ws-authorization. (http://msdn.microsoft.com/ws-security/)
17. Neumann, G., Strembeck, M.: An Approach to Engineer and Enforce Context Constraints in an RBAC Environment. In: Proceedings of the eighth ACM symposium on Access control models and technologies, ACM Press (2003) 65–79
18. Freier, A., Karlton, P., Kocher, P.: The SSL Protocol Version 3.0. Technical report, Netscape (1996)