"Policy-Maker", a Toolkit for Policy-Based Security Management

Andreas Pilz
Institute for Data Processing, TU München
andreas.pilz@ei.tum.de

Abstract

"Policy-Maker" is an implementation of our concept for the security management of heterogeneous networks. It is entirely based on the Common Information Model (CIM) and the Web-Based Enterprise Management (WBEM) architecture, which is an industry standard of the Distributed Management Task Force (DMTF).

In our concept an administrator can specify security policies uniformly and directly within the CIM data model via a comfortable grapical user interface (GUI) provided by our "Policy-Editor". The policies are processed and executed within the WBEM architecture, in which component specific "providers" map the policies to the mechanisms of the target network devices.

A policy can represent a hierarchy of rules, which are handled and solved in our concept by using several hierarchic provider-calls within the WBEM framework.

Furthermore, CIM-based policy models for some concrete security mechanisms (e.g. IP-Firewalls) have been designed and implemented for testing the "Policy-Maker".

1. Introduction

New information and communication services are using different network technologies, which are termed in this paper as "heterogeneous networks". Besides the traditional internet technologies for LAN and WAN[1] more and more mobile networks (e.g. GSM, UMTS[2]) and radio networks (e.g. IEEE 802.11) are used.

There are rising security requirements for providing new information and communication services, e.g. confidentiality, authenticity and integrity, access control and availability. The security requirements can only be met by comprehensive, uniform and integrated security management, which configures all participating networks and devices. Therefore security management of heterogeneous networks is a prerequisite, in order to achieve the demanded level of security for any information and communication service.

Management of heterogeneous networks is the configuration of network devices with different functionalities and interfaces, based on various technologies and on different manufacturers. The increasing demand for mobile services leads to dynamic scenarios, in which the configuration of the security mechanisms have to be permanently adopted by the actual policies. A security management system must support these dynamic scenarios.

Security management is a part of network and system management. Traditional management architectures are too simply structured for handling the problems mentioned above (e.g SNMP[3]) or have not proved successful for policy-based network management (e.g. TMN[4]). A policy within our context is a set of rules or a hierachy of rules, which consist of conditions and associated actions. Simple policies can directly be converted (e.g. by a single entry in an IP filter). Policies of a higher abstraction layer can affect mechanisms of several network components.

Several languages for specifying policies have already been developed, e.g. IBM's TPL[5] or Lucent's PDL[6]. These languages are limited to the specification of policies. There are no mechanisms for the conversion of policies into device dependent security mechanisms specified. Ponder [1,2] is another approach for policy-based network management. Within the Ponder toolkit policies are specified at a high-level of abstraction, then broken down internally into simple rules. Finally, the policies are compiled and mapped to rules for the network devices, using e.g. SNMP or CIM/WBEM architecture.

Within our "Policy-Maker" [6] concept the policies for the security management are directly specified within the CIM model [4] and executed within the WBEM architecture [5] as specified by the Distributed Management Task Force DMTF [3]. Thereby, policies of a higher abstraction layer are automatically broken down

[1] LAN: local area network, WAN: wide area network
[2] GSM: global system for mobile communication, UMTS: universal mobile telecommunication system
[3] SNMP: Simple Network Management Protocol
[4] TMN: Telecommunications Management Protocol
[5] TPL: Trust Policy Language
[6] PDL: Policy Definition Language

into simple rules within the WBEM by hierachical provider calls. An external policy preprocessing is not necessary. The "Policy-Editor" offers a graphical user interface (GUI) for editing the policy objects.

In this paper we present our concepts and the "Policy-Maker" implementation. Finally, we compare the Ponder concept with our Policy-Maker.

We assume that the reader is familiar with the Common Information Model CIM and Web-Based Management architecture WBEM of the DMTF. Relevant information on these management concepts can be found in [3,4,5].

2. Policy-Maker

Our security management system "Policy-Maker" is based on the CIM model [4] and the WBEM management architecture [5] of the DMTF [3].

We extended the CIM model by models for some specific security mechanisms, e.g. IP and CORBA firewalls and Intrusion Detection Systems (IDS). In contrast to other policy-based approaches we integrated the "intelligence" for the policy conversion and transformation directly into the WBEM architecture by using special providers. In particular we present our concepts for creating policy trees and the transformation of these trees. The "Policy-Editor" provides a graphical user interface (GUI), which allows for specifying policies and rules in a comfortable way for the administrator. The GUI is automatically generated based on the information stored in the repository of the WBEM. All associations and aggregations of the policy objects are automatically handled by the "Policy-Editor". Therefore, the administrator can concentrate on specifying the policies in a graphical tree representation.

2.1 CIM Model Extensions

At the beginning of the concept development phase there existed a CIM model for IPSec policies. We developed CIM models for some further security mechanisms, e.g. IP and CORBA firewalls and IDS [6]. We presented our models to the DMTF in January 2003.

The CIM models of firewalls were the basis for implementing and testing our concepts.

2.2 Policy Transformation

Real network devices depend upon their proprietary management capabilites. Within the WBEM architecture, providers perform the conversion, transformation and translation of the generic management information for real network devices with their proprietary management capabilites. We are using the provider mechanisms not only for "direct" transformations on real network devices, but also for multi-level or "indirect" transformations within a WBEM or between different WBEMs. In the following, policies which can be "directly" transformed are named "direct policies". In contrast to that, there are the so-called "indirect policies", which can not be directly transformed and processed by a network device. An "indirect policy" is a tree, consisting of "indirect policies" and finally "direct policies".

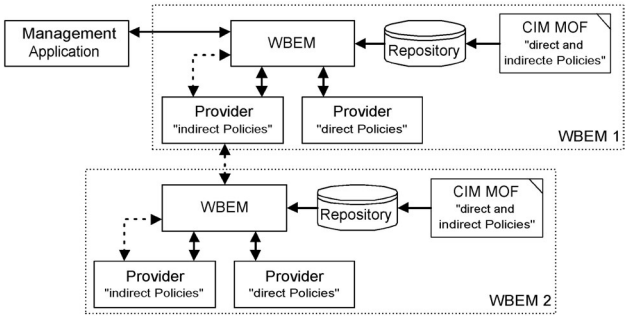

Figure 1: Processing of "direct" and "indirect policies"

In Figure 1, the extended provider functionality is shown. There are two WBEM systems (WBEM1 and WBEM2). Both WBEM systems have their own repository and providers. A "direct policy" can be handled e.g. within WBEM1 using a "direct provider" (e.g. provider for firewalls). An "indirect policy" is handled by the corresponding provider for "indirect policies" (here: the "indirect provider" of WBEM1 performs management actions on WBEM1 and WBEM2). This provider breaks the "indirect policy" down into its next sub-policies. These sub-policies can be "indirect" or "direct policies". The "direct sub-policies" can be handled by the "direct providers", the "indirect sub-policies" are further broken down to their sub-policies. This is done, until all "indirect policies" are broken down entirely to their "direct policies", which can be directly applied to the real devices. In contrast to other policy-based management approaches, the whole policy processing, conversion and translation and therefore the "intelligence" of the system can be integrated in an existing WBEM system without any modification of the WBEM implementation. Only the CIM models, which

are necessary for the policy processing and the providers have to be installed.

We developed a policy template approach, which we implemented the policy transformation concept as a prototype. An administrator can define new policies, which are composed out of other policies. Therefore, several policies can be summarized into one single policy and hierarchical policy trees can be created, as Figure 2 shows.

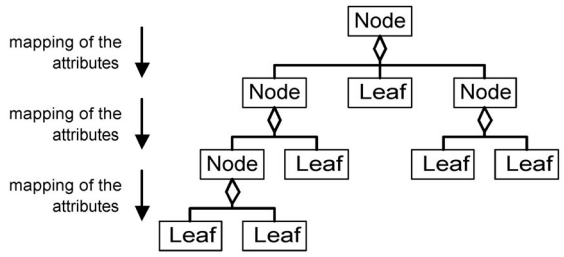

Figure 2: Policy Tree - Example

The administrator creates this tree preferably bottom-up. He or she starts with the leaf object, which corresponds to “direct policies”. The leaf object can be combined by using node objects in the next abstraction layer. The node objects correspond to “indirect policies”. The node objects can be summarized by superior node objects again.

The policy tree is processed top down. In the first step the root node is processed by a special provider, which maps the attributes of the root node to the attributes of the inferior node objects. These nodes are in turn processed by the corresponding providers. This is done, until the attributes of the root node are mapped to all leaf nodes of the policy tree. The leaf nodes correspond to “direct policies” and can therefore be mapped to the target devices.

This concept simplifies the reuse of policy blocks, in which only a small amount of attributes has to be changed. Our concept provides two possibilites for the transformation of such policy trees with special classes and providers or with universal ones, respectively.

1. **special node and leaf classes with special providers**
   In case of special node and leaf classes, the transformation information is integrated into the provider. Therefore, a special provider for each class is needed. Using this approach, an administrator has to implement special node and leaf classes as well as special providers.
2. **universal node and leaf classes with universal providers**
   In this case, the transformation information can directly be integrated into the node and leaf objects by using an XML description. The advantage of this approach is, that no special node and leaf classes or special providers have to be implemented. A universal provider can be used, which extracts the XML transformation description out of the corresponding node or leaf object. It performs the transformation according to the description. Therefore, an administrator has only to instantiate node and leaf objects and provide the transformation description.

2.3 Policy-Editor

Policy-Editor is an application for specification and modification of policies [6]. A GUI is dynamically generated out of the CIM model. A goal attribute of the Policy-Editor was to support any CIM model derived from the basic policy CIM model, whereas only a minimum set of rules should be hardwired. Moreover, all aggregations and associations of the objects should be handled automatically. In contrast to existing CIM-Browsers or CIM–Editors, the administrator is able to concentrate on specifying the policies. He or she does need to take care about the CIM internal management of objects and relations. In addition, type checking of the input of the administrator was extended using XML. Another important aspect is the provision of transactions, in order to guarantee a consistent state.

A multi-user system can be implemented within the WBEM. Thus, the Policy-Editor itself can be designed as a single-user application.

In Figure 3, a screenshot of the Policy-Editor is shown. The GUI is divided into two parts. On the left hand side the policy and policy group objects are presented in a tree-structure. Several operations can be performed on these objects, e.g. adding or deleting of aggregated or associated objects. The CIM model is analyzed thereby and only operations which are consistent with the CIM model are provided for the user.

On the right hand side, the components of the currently selected policy rule node are shown. In general, a policy consists of a condition, a time period and an action part.

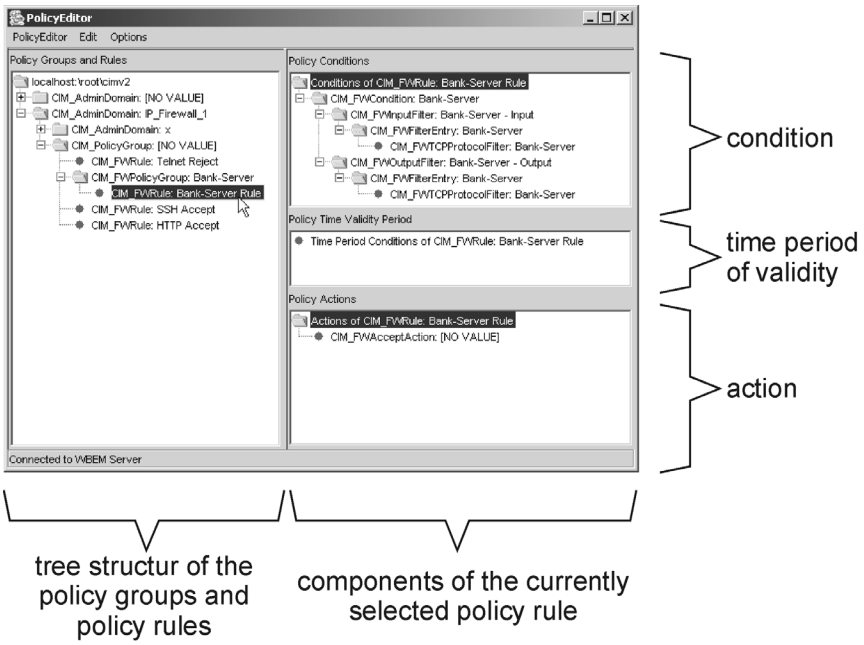

Figure 3: GUI of the Policy-Editor

In Figure 3 there is an example set of firewall rules defined for a banking application. On the left hand side the firewall policy object is currently selected. On the right hand side the components of the example firewall rule for the banking application are shown:

1. **Policy Condition**
   Here the condition part of the policy rule is defined. A tcp protocol filter for filtering certain ports and tcp flags in the input and output filter chain are defined.

2. **Policy Time Validity Period**
   In this part, the time period in which the rule is valid can be defined. The rule is always valid in this example, so no entry is needed.

3. **Policy Action**
   The action defined in this part has to be performed if the condition is true. In our example, the traffic should pass the firewall. Access from the outside to the bank server is allowed.

All objects and attributes can be modified. Each operation is performed immediately on the data stored in the repository of the WBEM server. Similarily, any change in the management data in the WBEM server takes effect on the Policy-Editor. For example if a new CIM model for policies of a new security mechanism is installed on the WBEM server, the Policy-Editor is able to deal with the new model without any modifications.

3. Comparison of Policy-Maker and Ponder

3.1 Ponder

Ponder [1,2] has been developed by the Policy Research Group of the Distributed Software Engineering Department of Computing at the Imperial College, London. It is a declarative, object oriented language for specifying security policies for various applications. Besides the policy language there is a ponder policy toolkit including the following components: a policy editor for the specification and modification of policies, a ponder compiler for the conversion and translation of the policy specifications to Java objects and a domain browser for a graphical representation of the structure of the domains.

Ponder is suitable for specification of policies. Even complex policies can be specified easily. The ponder rules and policies have to be mapped to the concrete target devices. There exists an approach, in which ponder policies are mapped to the CIM model.

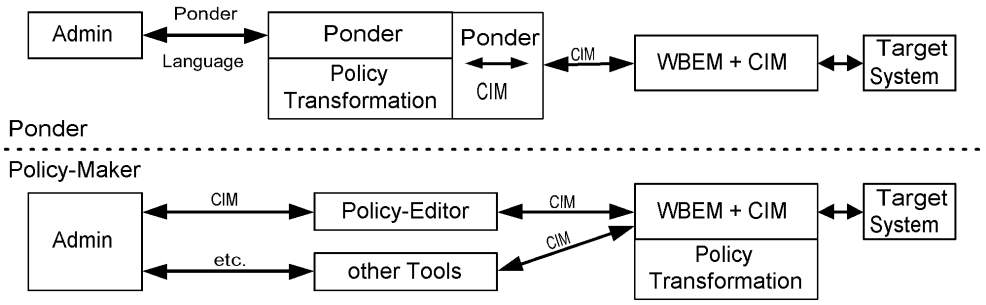

Figure 4: Comparison: Ponder – Policy-Maker

3.2 Comparison of Policy-Maker and Ponder

In Figure 4 the architecture of Policy-maker and Ponder is shown.

The functionality of both concepts is very similar. Within the Policy-Maker concept, the policy transformation and conversion is integrated into the WBEM and CIM. Within Ponder, policies are preprocessed in the first step within the ponder framework and after that the policies are mapped to the WBEM and CIM. Therefore, within the Ponder concept even little changes to the management information require a policy processing and an additional policy translation to CIM.

Within the Ponder framework, dealing with aggregations and associations is a subject of research. In contrast, the Policy-Editor of the Policy-Maker facilitates the work of the administrator by automatically handling all aggregations and associations.

The Policy-Maker concept has some more advantages: The Administrator does not need to learn a new language. Performing management operations directly in CIM results to a higher level of transparency. It is very easy to expand the concept by adding new providers to the existing system. The integration of the transformation intelligence is made very easy by using special classes and providers in the existing WBEM. Existing management tools based on CIM and WBEM can be used and integrated.

Policies have to be consistent and complete. There are still problems open to be solved within Ponder, Policy-Maker and any other existing policy-based management system, e.g. the detection and solution of conflicts, which are caused by contradictory policies. Another problem is the systematic policy refinement, because the network and security policies can not necessarily be applied to the security mechanisms and capabilities of existing systems.

4. Summary

The CIM model and the WBEM architecture of the DMTF are a good basis for an integrated security management system for heterogeneous systems. The standardized modelling of management information in CIM is a prerequisite for the management of heterogeneous system. The conversion and translation of the management information described in the uniform CIM model is processed by the so-called “provider”, which takes the system and manufacturer specific specialities into account.

Our Policy-Maker concept extends the existing CIM and WBEM of the DMTF. In particular, policies can be hierachically structured, whereas the leaves of the policy tree can be directly mapped to a concrete target device. Processing the policy tree is done within the WBEM system by hierarchical provider calls. The Policy-Maker concept has been implemented as a prototype. The Policy-Editor, the policy models for IP and CORBA firewalls, and the corresponding providers have been implemented, too.

5. References

[1] Ponder and Ponder Toolkit of the Policy Research Group of the Distributed Software Engineering Department of Computing at the Imperial College, London, UK: http://www-dse.doc.ic.ac.uk/Research/policies/index.shtml

[2] N.N. Damianou et al., „The Ponder Policy Specification Language“, Policies for Dist. Sys. Net., HP Labs Bristol, 29-31 Jan. 2001, pp. 18-38

[3] Distributed Management Task Force DTMF: www.dmtf.org

[4] Common Information Model CIM: http://www.dmtf.org/standards/standard_cim.php

[5] Web-Based Enterprise Management: http://www.dmtf.org/standards/standard_wbem.php

[6] Policy-Maker: http://www.ldv.ei.tum.de/page78