Policy Modeling and Refinement for Network Security Systems

João Porto de Albuquerque[1,2], Heiko Krumm[2], and Paulo Lício de Geus[1]

[1]Institute of Computing , State University of Campinas , 13083-970 Campinas/SP Brazil
[2]FB Informatik , University of Dortmund , 44221 Dortmund Germany

Abstract

In today's network environments the integrated design and management of different security technologies and mechanisms are of great interest. Especially in large networks, the security management should be supported by approaches with an appropriate level of abstraction, such that a system can be considered independently of the complex configuration details of its various component mechanisms. Furthermore, the employment of the security services and the design of their configurations should be supported by a structured technique that separates the consideration of the system as a whole from the detailed design of the subsystems. Pursuing these goals, this papers offers an approach to modeling network security systems, based on the concepts of policy-based management and model-based management, and analyzes the policy representation and refinement as well as the model validation enabled by this modeling.

1. Introduction

The current network environments incorporate an ever-increasing variety of security mechanisms in order to fulfill the protection needs against network-based attacks. In this complex scenario, the importance and costs of security management are growing rapidly. Therefore, automated tools and methodologies are of high interest to assist management tasks such as the installation and configuration of security services, as well as, during operation, their monitoring, auditing, adaptation and reconfiguration. Furthermore, the security management should be supported by approaches with an appropriate level of abstraction, such that the managed system could be considered independently of the heterogeneous configuration details of the manifold of security mechanisms and technologies used.

A prolific research branch in this direction is based upon the modeling of a system in terms of the *policies* that are applied to it. Policies are defined in this context as rules governing the choices in behavior of a system [7] and they can be specified in different abstraction levels. Thus, policy hierarchies [4, 9] can be built up, so that an initial set of high-level abstract policies can be refined through intermediate levels until reaching mechanically executable policies. These low-level policies could then be directly interpreted by policy-aware management agents (policy-based management [7]) or converted into configuration parameters for particular mechanisms.

There are a considerable number of approaches to policy specification both for security management and policy-driven network management purposes (see [8] for a survey). However, regarding the tool-assisted building of policy hierarchies and the automation of the policy refinement process, considerable research remains to be done.

The Model-Based Management (MBM) approach [2, 3] supports the building of policy hierarchies by means of an interactive graphical design. It adopts concepts of object-oriented system design and employs a model of the system that is vertically structured into a set of layers. The objects and associations of a layer represent the system to be managed and security policies on a certain abstraction level. A software tool is also provided to assist the modeling of system objects and policies, as well as to support automated policy refinement.

A problem of MBM occurs when dealing with large systems, since the representation of the policies and objects tends to lose much of its understandability, getting obscure due to the great number of elements. The Diagram of Abstract Subsystems (DAS) is proposed to improve the modeling technique of MBM, consisting of a new abstraction layer that offers a representation of the overall system structure segmented into *Abstract Subsystems*. In DAS, the details are hidden and dealt with in the internal specification of each subsystem, thereby improving the comprehensibility and scalability of the model.

In this paper, we analyze the policy support offered by MBM in comparison with a classification framework [8], and point out the extensions and improvements brought forth by the Diagram of Abstract Subsystems. Furthermore, this paper shows how DAS makes possible a model validation that performs examinations of the system's structural architecture in relation to the policies prescribed to it. As such, the modelled policy hierarchy can be validated according to consistency and completeness criteria applied to its successive abstraction levels.

For this purpose, in the following section we introduce MBM and in succession present the Diagram of Abstract Subsystems approach (Section 3). Subsequently, the policy support and refinement offered by DAS are detailed in Section 4 and the model validation is discussed in Section 5. Lastly, we cast conclusions for this paper in Section 6.

2. Model-Based Management

The Model-Based Management (MBM) approach [2, 3] aims to support policy-based management by the use of an object-oriented model of the system to be managed. Based upon this model, a policy refinement can be accomplished such that configuration parameters for security mechanisms can be automatically derived.

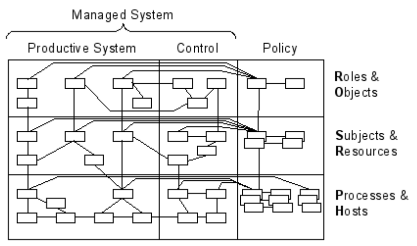

Figure 1. Model Overview

The structure of the model is shown in Figure 1, where three abstraction levels can be distinguished: *Roles & Objects* (RO), *Subjects & Resources* (SR), and *Processes & Hosts* (PH). Each level is a refinement of the superior level in the sense of a “policy hierarchy”[4]. The uppermost level (RO) offers a business-oriented view of the network whereas the lowest level is related to a technical view. The vertical subdivisions differentiate between the model of the actual managed system (with productive and control elements) and the policies that regulate this system. This last category encompasses requirement and permission objects, each of which refers to the model components of the same level and expresses security policies.

The uppermost level (RO) is based on concepts from Role-Based Access Control (RBAC) [6, 1]. The main classes in this level are: *Roles* in which people who are working in the modelled environment act; *Objects* of the modelled environment which should be subject to access control; and *AccessModes*; i.e. the ways of accessing objects. The class *AccessPermission* allows the performer of a *Role* to access a particular *Object* in the way defined by *AccessMode*.

The second level (SR in fig. 1) offers a system view defined from the standpoint of the services that the system will provide, and it thus consists of a more complex set of classes. Objects of these classes represent: (a) people working in the modelled environment (*User*); (b) subjects acting on the user’s behalf (*SubjectTypes*); (c) services in the network that are used to access resources (*Services*); (d) the dependency of a service on other services (*ServiceDependency*); and lastly (e) *Resources* in the network.

The lowest level (PH) is responsible for modeling the mechanisms that will be used to implement the security services defined in SR. Therefore, PH will have even more classes than before, representing for instance the *Hosts*, with their respective network *Interfaces* and *Processes*. *ProtocolPermissions* allow the transition of packets between processes. Several other classes are also defined according to the supported mechanisms; they will not be mentioned here for the sake of brevity.

MBM also provides a support tool, which, at first, assists the user in the modeling of the system by means of a graphical editor with additional functions for the checking model-dependent constraints. Once the system modeling is finished, the tool performs an automatic refinement of the abstract security policies (in the RO level), through the intermediary levels (SR and PH), until achieving configuration files for the supported security mechanisms. The security policy support of MBM is described in the next section.

2.1. Policy representation, semantics and refinement

According to Sloman and Lupu [8], policies can be sorted into two basic types: *authorization* and *obligation policies*. Authorization policies are used to define access rights for a subject (management agent, user, or role) and can be either *positive* (defining the actions subjects are *permitted* to perform on target objects) or *negative* (specifying the actions subjects are *forbidden* to perform on target objects). As such, authorization policies are used to define access control rules implemented by several types of mechanisms in a network security system, such as packet filters, Kerberos, and VPNs.

Obligation policies are, in turn, event-triggered condition-action rules that can be used to define the activities subjects (human or automated manager components) must perform on objects in the target domain; i.e. the *duties* of these subjects. In the network security context, obligation policies can be used to specify the behavior of mechanisms such as logging agents, intrusion detection systems (IDS) and watchdogs.

In MBM, authorization policies are represented by means of *AccessPermission* objects (see sec. 2) in the uppermost (RO) level. The set of *AccessPermissions* is given by the modeller and acquires in this context a double meaning: on the one hand it defines the explicit permission for *Roles* to access *Objects* (in the way defined by an *AccessMode*)—corresponding to the positive authorization policies in the above classification. On the other hand, the semantics of MBM also imply that all triples of *Role*, *Object* and *AccessMode* not belonging to the set of *AccessPermissions* are implicitly forbidden—i.e. it defines implicitly the negative authorization policies that must as well be enforced by the security mechanisms.

These authorization policies are, in MBM, subject to an automatic refinement when descending the abstraction levels of the model. In this manner, each one of the *AccessPermissions* is refined into one or more *ServicePermissions* in the SR level, which expresses an authorization for a *SubjectType* (on behalf of a *User*) to use a *Service* to access a *Resource*. Subsequently, the *ServicePermissions* are refined into a set of *ProtocolPermissions* at the PH level, which in turn, for the last step of model-based security service configuration, are evaluated by a series of backend modules, in order to generate the adequate configuration files for each of the supported security service products.

It should be noted that each permission set derived from the given *AccessPermissions* (i.e. *ServicePermissions* and *ProtocolPermissions*) stands also for both explicit positive and implicit negative authorization policies. This property must be taken into consideration during the whole process of automatic policy refinement described above.

As for obligation policies, these are not directly represented in MBM since the modeling used in MBM builds upon RBAC (see sec. 2), which in its basic form does not enclose obligation policies (referred to as duties in [6]). However, as a further development of MBM, besides the above elements, the RO level also incorporates prescriptive *SecurityVector* objects [5]. These *SecurityVectors* offer an extension to the RBAC model in order to represent the *security levels* that must be enforced in the context of an *AccessPermission*. The security levels are defined by the modeller and represented by a vector of desired values (natural numbers from 1 to 5) for the *security requirements* of confidentiality, integrity, availability and traceability.

Additionally, both the SR and PH levels also have *SecurityVectors*. These objects also consist of a vector of security levels, but in this case the values describe what can be *provided* by the entity of the model to which they are assigned; they are thus called warranting *SecurityVectors*. As such, warranting *SecurityVectors* are associated to each *Service* in order to represent the security levels that it is able to warrant. In the PH level, warranting *SecurityVectors* are associated to *Trust Areas* that represent groups of PH-elements that are able to warrant the same security levels.

During the process of policy refinement described above, the security levels warranted by mechanisms are checked against the security requirements prescribed in the *SecurityVectors* of the RO level. As such, a *SecurityVector* with a high level of confidentiality could determine, for instance, the decision between using an encrypted tunnel instead of plain text. Another practical example occurs when a security mechanism that supports logging has this functionality activated in order to fulfill the high traceability level required by the prescriptive *SecurityVector* of its corresponding *AccessPermission*. The examples show that, though *SecurityVectors* do not directly represent obligation policies, the analysis of the security requirements they express can yield configuration parameters for mechanisms that do correspond to the "need to do" aspects in the system, and hence to obligation policies.

3. Diagram of Abstract Subsystems (DAS)

From the previous discussion of MBM, one can notice that the complexity of the PH level—which shall depict the entire system, with its processes, hosts, network interfaces etc.—increases rapidly as the size of the modelled system grows. Consequently, models of large real systems tend to become quite confusing. In order to overcome this problem, a new layer has been introduced in the modeling: the Diagram of Abstract Subsystems (DAS).

DAS is a layer located immediately below the service-oriented view of the system (SR level in fig. 1) and above the PH layer. Its main objective is to describe the *overall structure* of the system in a modular fashion; i.e. to cast the system into its building blocks (ASs) and to indicate the connections between them.

Therefore, a DAS is a graph comprised of Abstract Subsystems (ASs) as nodes, and with edges that represent the possibility of bidirectional communication between two ASs. An AS, in turn, is an abstract view of a system segment; i.e. a simplified representation of a given group of system components. As such, an AS is itself a subgraph of a DAS and may contain the following types of elements:

Actors: groups of individuals in a system which have an *active* behavior; i.e. they initiate communication and execute mandatory operations according to *obligation policies*.

Mediators: elements that intermediate communications, receiving requests, inspecting traffic, filtering and/or transforming the data flow according to the *authorization policies*; they can also perform mandatory operations based on *obligation policies*, such as registering some information about data flows.

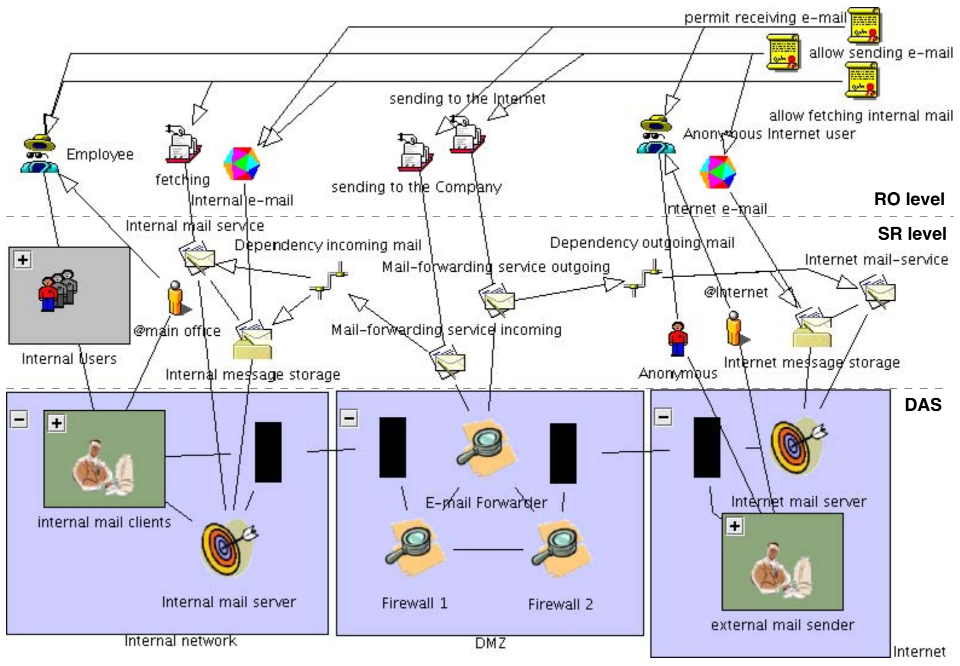

Figure 2. Three-layered Model

Targets: *passive* elements; they contain relevant information, which is accessed by *actors*.

Connectors: represent the interfaces of one AS with another; i.e. they allow information to flow from, and to, an AS. scalability

Each one of these element types represent a group of entities in the actual system, e.g. hosts, processes, protocols, network interfaces, network connections. The objects in DAS thus represent aggregates of such entities, grouped according to a policy-oriented view of the system, in order to present only the relevant aspects for a global view of the system structure.

A model example is shown in Figure 2, in which a DAS (at the bottom) is represented together with the levels RO and SR. This model represents a typical network environment, for which three *AccessPermissions* are defined at the uppermost level (RO), in order to regulate the access rights of the users in the internal network with respect to e-mail transactions, as well as to allow the users to receive e-mails from the Internet. At the bottom of this figure, the DAS for this environment illustrates the concepts previously explained in this section.

To complete the modeling, each AS in a DAS is expanded into a detailed view of the actual mechanisms of the system; i.e. the PH level. Figure 3 shows the PH level for the AS "internal network" as an example. Comparing the simplified view (in the DAS of fig. 2) and the detailed one (fig. 3), it can be observed that modeling through abstract subsystems offers concrete advantages in the conciseness and understandability of the model, as well as providing an intelligible view of the system architecture.

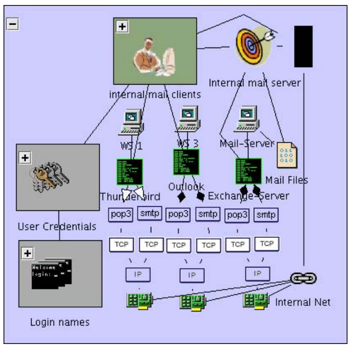

Figure 3. Expanded AS

Aside from the aforementioned modeling improvements, the DAS is also advantageous to policy support and model validation. These topics will be presented in the following sections.

4. Policy Support and Hierarchy

Along with the ASs, DAS also has objects to represent the security policies applied to the elements of the system.

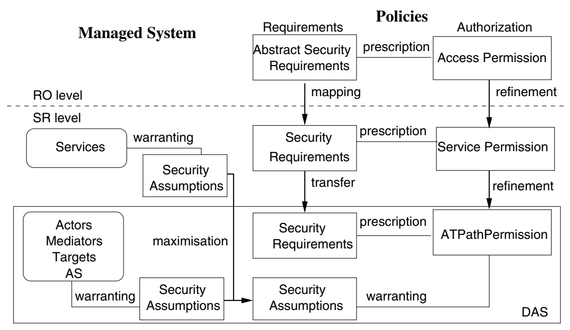

Figure 4. Policies and Security Requirements in the System

An overview of the hierarchical structure of these policy types is presented in Figure 4 with special emphasis on DAS (several details of the other levels were omitted). In the right hand columns of this figure, one can notice that the policy support is subdivided in two parallel tracks, corresponding to two policy types: *authorization policies* and *security requirements*. These tracks are discussed in more detail below.

The authorization policies in DAS are represented by a set of *ATPathPermission* objects, which are not directly modelled by the user, but rather automatically generated from the *ServicePermissions* and their related objects of the SR level (see sec. 2.1). Each *ATPathPermission* is a path in the graph and represents the permission for an *Actor* (initial node) to reach a certain *Target* (final node) passing through the required *Mediator* and *Connector* nodes. As such, an *ATPathPermission* is the abstraction refinement of a *ServicePermission* (see sec. 4).

Furthermore, there are also two types of *SecurityVector* (see sec.2.1) objects present in the DAS that represent security requirements. The first type consists of a set of *prescriptive SecurityVectors* (henceforth called *SecurityRequirements*), each one of which is related to a *Target* in order to specify the required security levels for this entity. The *SecurityRequirement* for a *Target* is calculated from the ones that are associated to the corresponding *Service* and *Resource* objects (in the SR level). These, in turn, are mapped from the abstract prescriptive *SecurityRequirement* vector attributed by the modeller to the corresponding *Object* and *AccessMode* in the RO level (see fig. 4).

The second type of *SecurityVector* represents the security levels that can be *assured* by elements in DAS. For each *ATPathPermission*, a detailed *warranting SecurityVector* (henceforth called *SecurityAssumption*) is calculated as the maximum security level that is assured by the *Services* that take part in the corresponding *ServicePermission*. Each of these *Services* has itself a related *SecurityAssumption*, representing the security levels that it is capable to provide. This security level determination models situations in which a certain *Service* (with particularly desirable security properties) can be utilized in order to improve the security level of a *ServicePermission*.

SecurityAssumptions have also to be manually assigned by the modeller to each *Actor*, *Target* and *Mediator* in a DAS, according to the security levels that can be warranted by the corresponding mechanisms in the PH level. If an element of these types does not have an associated *SecurityAssumption*, the minimum value for each security level is assumed. In order to offer a more compact representation of these relationships, an AS as a whole can also be associated to one *SecurityAssumption* in order to represent contextual information; i.e. this association would have the meaning that all of the elements inside this AS share certain security properties. This is useful, for instance, to model the fact that an internal network should have a higher confidentiality level than a public network.

5. Abstraction Refinement and Model Validation

Although the fully automated derivation of low-level, executable policies from a set of abstract specifications is, in the general case, not practical [8, 9], our modeling technique makes possible an automation of the building of a policy hierarchy on the basis of a system's model that is structured in different abstraction levels. In this process, the analysis of the system's objects, relationships and policies at a certain abstraction level enables the generation of lower level policies, based also on the system's model in the lower level and on the relations between entities of the two layers. As such, the model entities of a certain level and their relationships supply the contextual information needed to automatically interpret and refine the policies of the same level.

To be practically useful, however, the validation criteria of *consistency* and *completeness* must be assured between the different model layers. We understand that two policy sets pertaining to adjacent abstraction levels are *consistent* when there are no conflicts both inside the generated lower-level set as well as in relation to the specified high-level policies. In the context of our modeling, this implies that all accesses enabled by lower-level policies are correspondingly allowed by the given higher-level policy set. On the other hand, the *completeness* of the refinement is established when *all* higher-level policies are effectively enforced by the lower-level set.

These criteria should be verified on each step of the policy refinement that was described in the previous section. In our model, the validation starts at the SR level and verifies the aforementioned criteria in the refinement from this level to the DAS, and from the latter to the PH-level models. As such, the validation is performed in two phases. The first phase will have as scope the whole system, and its goal will be to apply the above criteria to the refinement from the service-oriented view of the system (in which a policy consists of a permission for a user to employ a service to access a resource) to its structural representation in a DAS (whose polices are enabled paths in the network by means of which an actor is allowed to access a target).

The second phase, in turn, has a local scope and aims at checking whether the abstract view of each subsystem (AS) corresponds to the subsystem elements in the PH level. Figure 5 depicts the validation scheme of the second test phase. This figure shows that the local conditions LC1 and LC2 are verified in each subsystem, in order to compare its abstract representation in DAS with the detailed information of the network mechanisms in the PH level. The assumptions made between the interconnection of the subsystems (CA1) and the configuration of the PH mechanisms (CA2) are also represented in Figure 5.

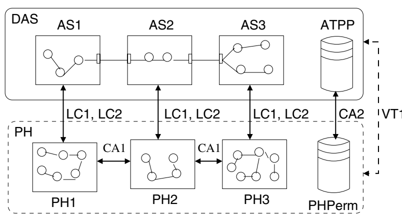

Figure 5. Validation Scheme

Further, we prove in general (by means of theorem VT1) that if the local conditions are asserted and our composition assumptions hold, then the whole PH level is valid in relation to the DAS.

The following sections will be dedicated to examining each one of these phases and then to analyze the whole validation process. For this purpose, in the next section we first present a formalization of the relevant model entities.

5.1. Model Formalization

Since the validation starts at the SR level, we first consider the entities of this level. The basic object types (see sec. 2) are thus formally defined by means of the disjoint sets: Usr encompassing the *User* objects of the system; Sty for the *SubjectType* objects; Srv for the *Services*; Rsc for the *Resources*.

Additionally, the relations between objects are defined as follows:

- $SrvDep \subset Srv \times Srv \times Rsc$, to express that a service depends on another to access a resource;
- $SP \subseteq Usr \times Sty \times Srv \times Rsc$, representing *ServicePermission* objects (see sec. 2);

In order to facilitate the model validation of the next sections, we define, along with the set of *positive* authorization policies SP, a set of *negative* authorization policies $\overline{SP}$, which is complementary to the former one. This second set is due to the double semantic of *ServicePermissions* in our modeling (see sec. 2.1).

The security requirements, in turn, are represented by the function srq, which gives the prescriptive *SecurityVector* associated to *Services* and *Resources* (see sec. 4): $srq(x : Srv \cup Rsc) \rightarrow SecVector$.

For the DAS level, the following basic sets are defined: Act comprising the *Actor* objects; Med for *Mediators*; Tar for *Targets*, $Conn$ for *Connectors*; and AS for ASs. DAS itself is hence defined as the graph $DAS = (V,E)$, where $V := Act \cup Med \cup Tar \cup Conn$ and E is a set of edges that connect the nodes in V. Further object relations are represented by:

- $RAct \subset Act \times UsrSty$, encompassing abstraction refinements from a pair of *User* and *SubjectType* objects to an *Actor*;
- $RMed \subset Med \times Srv$, for refinements from *Services* to *Mediators*.
- $RTar \subset Tar \times SrvRsc$, representing refinements from *Service* and *Resource* pairs to a *Target*;
- $as : V \rightarrow AS$, a function that maps the association of nodes in V to abstract subsystems in AS.

The authorization policies in this level are represented by ATPP objects (Actor-Target Path Permissions, see sec. 4), defined as paths in the graph of the form:

$$\begin{aligned} ATPP \subset \{\langle v_1 \rightsquigarrow v_k\rangle : \quad & v_1 \in Act,\ v_k \in Tar, \\ & v_2,\dots,v_{k-1} \in Med \cup Conn, \\ & \big(v_j, v_{j+1}\big) \in E \quad (1 \le j < k)\} \end{aligned}$$

(We shall henceforth use the symbol $\rightsquigarrow$ to denote paths in a graph.)

The abstraction refinement relation from a service permission to an ATPP is represented by the set $RATPP \subseteq ATPP \times SP$.

In order to represent the security levels that can be assured by individual elements in DAS (see sec. 4), a function sa is defined as $sa(x) : x \rightarrow SecVector$; where x pertain to one of the sets: AS, V or $ATPP$.

The PH level, in turn, is also represented by a graph $PH := (V_{PH}, E_{PH})$. Since this level contains a series of component types that are not relevant for the validation bellow, they will not be mentioned here. On the other hand, the following relation sets are defined:

- $RPHAct \subset V_{PH} \times Act$, representing the refinements from *Actors* to elements of the PH level;
- $RPHTar \subset V_{PH} \times Tar$, for refinements from *Targets* to PH level nodes;
- $RPH \subset V_{PH} \times V$, representing the general refinements from components of DAS to PH level nodes;
- $as : V_{PH} \to AS$, overriding of function *as* to map the association of PH nodes to abstract subsystems in *AS*.

The set of *PHPermission* objects represent the authorization policies in this lowermost level. This permissions are also paths in the PH-level graph defined as:

$$PHPerm \subset \{\langle v_1 \rightsquigarrow v_m \rangle : \quad v_1, \ldots, v_m \in V_{PH}, \\ (v_j, v_{j+1}) \in E_{PH} \quad (1 \le j \le m)\}$$

Along with the above model's entities, we define a predicate that reflects the functioning of the system:

$$access_{PH} : V_{PH} \times V_{PH} \to boolean$$

$$access_{PH}(v_1, v_2) = \begin{cases} 1 & \text{if element } v_1 \text{ can access } v_2 \\ 0 & \text{otherwise} \end{cases}$$

Having completed our definitions, we now proceed to the first validation phase.

5.2. Global Conditions and Assumptions

The first global check that is performed in the model certifies that: (GC1) for each *ServicePermission* a *corresponding ATPathPermission* has to be found in the DAS. This means that all the security policies in the SR level are *structurally feasible* in the DAS; i.e. that the authorization policies prescribed in the SR level are effectively enabled by its corresponding policies in the DAS.

The *correspondence* relation, used in the latter condition, between a *ServicePermission* and an *ATPathPermission*—which will be referred to by *sp* and *atpp*—is intuitively satisfied when: (i) the *Actor* of *atpp* is related to the *User* and *SubjectType* of *sp*, (ii) the *Target* of *atpp* is related to the *Service* and *Resource* of *sp*, and (iii) each *Service* in the service dependencies of *sp* is related to a *Mediator* of *atpp*. Furthermore, the compliance of *atpp* with the security levels prescribed by *SecurityRequirement* vectors (see sec. 4) must be verified. This compliance is established if: (iv) each of the elements in *atpp* (*Actors*, *Mediators* and *Targets*) is able to provide security levels equal to or greater than the maximum between the values prescribed by the *SecurityRequirements* of the corresponding *Service* and *Resource* objects (this comparison is performed on the numerical value attributed by the modeler to each dimension of the involved security vectors; it is further elaborated in [5]).

With these considerations, we can formally define CG1 as follows:

Global Condition 1 (GC1)

$$\begin{aligned} &\forall\, sp(usr, sty, srv, rsc) \in SP \\ &\quad \exists\, atpp\,\langle v_1 \rightsquigarrow v_m \rangle \in ATPP : \\ &\qquad (sp, atpp) \in RATPP \wedge\ (v_1, usr, sty) \in RAct \\ &\qquad \wedge\ (v_m, srv, rsc) \in RTar \\ &\qquad \wedge\ \forall\, srvd : \exists\, (srvd, srv, rsc) \in SrvDep \Rightarrow \\ &\qquad\qquad \exists\, v_j : (v_j, srvd) \in RMed, \quad (1 < j < m) \\ &\qquad \wedge\ min_{q=1}^{q \le m} \quad esa[v_q, atpp] \ge\ max(srq[srv], srq[rsc]) \end{aligned}$$

The effective security levels that can be assured by each node of the DAS (given by the function *esa* above) are, in turn, calculated as the maximum between the values assured by the node itself (represented by its corresponding *SecurityAssumption*), the security levels associated to the AS to which the node belongs, and the *SecurityAssumption* associated to the *ATPathPermission* (see sec. 4). In this respect, the maximization of the values reflects the result of the collaboration of different mechanisms and context information to the effective security level achieved by an individual in the system. The auxiliary function *esa* is thus defined as:

$$esa(v : V,\ atpp : ATPP) = \\ max(sa[v], sa[as[v]], sa[atpp])$$

Subsequently, it should be asserted that: (GC2) for each negative authorization policy in the SR level (i.e. elements of the auxiliary set $\overline{SP}$), all the paths between its related *Actor* and *Target* objects in the DAS must be disabled. As such, *only* the paths that are allowed by security policies in the SR level will enabled in the DAS. This condition can be formally expressed as:

Global Condition 2 (GC2)

$$\begin{aligned} &\forall\, \overline{sp}\ (usr, sty, srv, rsc) \in \overline{SP} \\ &\quad \forall\, (act, usr, sty) \in RAct,\ \forall\, (tar, srv, rsc) \in RTar \\ &\qquad \nexists\, p\,\langle v_1 \rightsquigarrow v_n \rangle : \quad v_1 = act,\ v_n = tar, \\ &\qquad\quad (v_i, v_{i+1}) \in E\ (1 \le i < n),\ enabled(p) \end{aligned}$$

Along with this last condition, we assume that all the DAS elements are configured in such a way that they only enable communication corresponding to an *ATPathPermission* to pass through. This particularly concerns *Mediators*, which are presumed to disable all the paths passing through them that are not allowed by an element of *ATPP*. This premise reflects the *correct implementation* of the mechanisms and is formalized in the axiom GA1:

Global Axiom 1 (GA1)

$$\forall\, p\,\langle v_1 \rightsquigarrow v_n\rangle :\ v_1,\ldots,v_n \in V,\ (v_i,v_{i+1}) \in E\ (1 \le i < n)$$
$$enabled(p) \Leftrightarrow \exists\ atpp \in ATPP :\ p \subseteq atpp$$

Therefore, the completeness of the abstraction refinement SR level $\rightarrow$ DAS results from GC1 and GA1, since for each $sp \in SP$ there is a implementing $atpp \in ATPP$ (GC1), which in turn has a corresponding enabled path in DAS (GA1); i.e. for each service permission in the SR level there is an enabled path in DAS. The consistency criterion is then satisfied by the assurance of GC2, for it implies that only the paths corresponding to service permissions of the SR level are enabled in DAS.

5.3. Local Conditions

In a second phase, in order to verify whether each PH subsystem is a valid refinement of the respective AS, checks are performed that are analogous to the ones previously described, but restricted now to the scope of a particular subsystem. The first test, similar to GC1, certifies that: (LC1) for each structural connection between two elements in an AS, say A and B, a PH-model path must exist between each object that refines A and each object that refines B.

Local Condition 1 (LC1)

$$\forall\ (v_1^{AS}, v_2^{AS}) \in E$$
$$\exists\ p\ \langle v_1^{PH} \rightsquigarrow v_n^{PH}\rangle :\quad v_1^{PH},\ldots,v_n^{PH} \in V_{PH},$$
$$(v_i^{PH}, v_{i+1}^{PH}) \in E_{PH}\ (1 \le i < n),$$
$$(v_1^{AS}, v_1^{PH}) \in RPH,\ (v_2^{AS}, v_n^{PH}) \in RPH$$

Then, following on in analogy to GC2, it should be attested that: (LC2) there exists no "not allowed" connections between PH-model objects. As regards the DAS, however, we do not have a set of negative authorization policies in order to determine the invalid connections, as in the SR level. Nevertheless, both levels have the double semantics of explicit positive and implicit negative authorization policies. That implies all the disabled paths in a DAS represent forbidden communication.

Therefore, in order to assure the compliance of each PH subsystem to its corresponding AS in this respect, we assert that for all paths between two PH level objects, say C and D, which are related to objects in the AS, say E and F, there must be an enabled path in the AS between E and F.

Local Condition 2 (LC2)

$$\forall\ p\ \langle v_1^{PH} \rightsquigarrow v_n^{PH}\rangle :\quad v_1^{PH},\ldots,v_n^{PH} \in V_{PH},$$
$$(v_i^{PH}, v_{i+1}^{PH}) \in E_{PH}\ (1 \le i < n),$$
$$(v_1^{PH}, v_1^{AS}),\ (v_n^{PH}, v_m^{AS}) \in RPH$$
$$\exists\ q\,\langle v_1^{AS} \rightsquigarrow v_m^{AS}\rangle :\ v_1^{AS},\ldots,v_m^{AS} \in V,$$
$$(v_j^{AS}, v_{j+1}^{AS}) \in E \quad (1 \le j < m),\ enabled(q)$$
$$\wedge\ \forall\ v_k^{AS}\ (1 < k < m) \Rightarrow$$
$$\exists\ (v_l^{PH}, v_k^{AS}) \in RPH\ (1 < l < n)$$

5.4. Composition Assumptions

Besides the local conditions, an assumption must be made about the interconnection between the PH subsystems. In the DAS, the connection between two ASs is always performed through a pair of *Connector* objects (one in the first AS and another on the second AS). These *Connectors* are abstract *passive* entities that are refined into PH-level objects representing communication devices, e.g. ethernet networks, network interfaces (see fig. 3 for an instance). Since the models in the PH level are segmented into ASs (i.e. each AS has a separate PH model as depicted in fig. 5), the interconnection between PH-level nodes that refine *Connectors* is not represented in these local diagrams. Nevertheless, an enabled PH-level connection between two objects of different PH subsystems (corresponding to different ASs) is assumed to be present if, and only if, there is a corresponding structural connection in the DAS. This assumption is defined by the axiom CA1:

Composition Axiom 1 (CA1)

$$(v_1^{PH}, v_2^{PH}) \in E_{PH} :$$
$$as[v_1^{PH}] \neq as[v_2^{PH}]$$
$$\Leftrightarrow \exists\ (v_1^{DAS}, v_2^{DAS}) \in E :$$
$$(v_1^{PH}, v_1^{DAS}),\ (v_2^{PH}, v_2^{DAS}) \in RPH,$$
$$as[v_1^{DAS}] = as[v_1^{PH}],\ as[v_2^{DAS}] = as[v_2^{PH}]$$

As in the DAS, it is also necessary to make considerations about the configuration of the mechanisms of the PH level. Our assumption here is that all security mechanisms and equipments in the PH level will be correctly and as strictly as possible configured; i.e. each PH-level object must restrict the information flow, as effectively as it is able to, in order to allow *only* the communication corresponding to what is enabled by its related element in the AS. Since the DAS objects are assumed to be configured according to the *ATPathPermissions* of the system (GA1), in order to formalize this assumption we use the set *PHPerm* of *PH-Permissions*, each one of which is an enabled path in the

PH level related to one *ATPathPermission*. We state then that for each possible access between two PH objects (represented by the function $access_{PH}$) there must be a related element in *PHPerm*:

Composition Axiom 2 (CA2)

$$access_{PH}(v_1, v_2) \Leftrightarrow \exists \ \langle v_1 \rightsquigarrow v_2 \rangle \subseteq p \in PHPerm$$

Notice that, analogously to GA1, this assumption also reflects the presumed correct implementation of the mechanisms.

5.5. DAS/PH Validation

In order to prove that the local conditions and assumptions are sufficient to validate the abstract refinement from the DAS to the PH level we must consider now these layers as a whole, thus achieving a generalized result from the previous considerations (as illustrated in fig. 5). Hence, we must demonstrate that, provided all the previous conditions and assumptions hold, an access, in the PH level, from a process to a system resource can be performed if, and only if, it is allowed in DAS; i.e. iff the corresponding abstraction of the process (*Actor*) is permitted to access the abstraction of the resource (*Target*) by means of an *ATPathPermission*. Theorem VT1 formalizes this assertion (the abstraction relations are respectively mapped by the auxiliary functions A and T).

Validation Theorem 1 (VT1)

$$access_{PH}(p, o) \Leftrightarrow \exists \ \langle A[p] \rightsquigarrow T[o] \rangle \in ATPP$$

The proof for the left to right direction of the theorem is based on an induction in the number of ASs for which the assertion is valid. For the basis, the two PH objects pertain to the same AS ($as[p] = as[o]$), and thus CA2 implies there is a path $\langle p \rightsquigarrow o \rangle$ in the PH model. Then, from LC2, and considering that these objects are related to their corresponding abstract entities in DAS ($a = A[p], t = T[o]$), this results in that there is a corresponding enabled path $\langle a \rightsquigarrow t \rangle$ in DAS. Now, GA1 implies this enabled path must be allowed by an *ATPathPermission*. Q.E.D.

In the general case, CA2 implies the PH-path that leads process p to system resource o spans n ASs and can be formally described as: $\langle v_1 \rightsquigarrow v_j, v_{j+1} \rightsquigarrow o \rangle$, where $v_1 = p$ and $as[v_1] = as[v_2] = \cdots = as[v_j]$ (they all pertain to the same AS). We first apply the basis case to the subpath $\langle p \rightsquigarrow v_j \rangle$, obtaining that it is allowed by some $atpp \in ATPP$. From CA1, results then v_j and v_{j+1} are correspondingly refinements from two connectors, say c_1 and c_2, such that there is a connection $(c_1, c_2) \in E$. Since connectors are always passive elements (see sec. 3), all enabled paths in the DAS that reach c_1 will be also capable of reaching c_2, and this implies (GA1) that $atpp$ also allows the path $\langle v_1 \rightsquigarrow v_j, v_{j+1} \rangle$; i.e. there is an enabled path $ep_1 \langle A[p] \rightsquigarrow c_1, c_2 \rangle$ in DAS. Now, for the subpath $\langle v_{j+1} \rightsquigarrow o \rangle$ the induction hypothesis can be applied (it spans $n-1$ ASs), resulting in that it is allowed by some *ATPathPermission* that thus contains the path $ep_2 \langle c_2 \rightsquigarrow T[o] \rangle$. The concatenation of ep_1 and ep_2 (eliminating the common element c_2) thus achieves an enabled path in the DAS $\langle A[p] \rightsquigarrow T[o] \rangle$, and GA1 implies in this case there is a related *ATPathPermission*. Q.E.D.

The opposite direction of the theorem asserts that for each *ATPathPermission* that allows a certain actor $A[p]$ to access a target $T[o]$, there must be a corresponding access possibility in the PH level. An induction in the number of ASs spanned by the *ATPathPermission* (said $atpp$) will be then used. For the basis, $atpp$ is local to an AS and, since all connections in $atpp$ are, by definition, also connections in the AS, the existence of a corresponding enabled path $\langle p \rightsquigarrow o \rangle$ in the PH subsystem is given by condition LC1. Thus, from CA2, this results in that this enabled path (which must be contained in a *PHPermission*) implies the access from p to o is possible. Q.E.D.

Assuming now that $atpp$ spans k ASs, it can be represented as a path of the form: $\langle A[p] \rightsquigarrow c_1, c_2 \rightsquigarrow T[o] \rangle$, where the subpath $\langle A[p] \rightsquigarrow c_1 \rangle$ is contained entirely in one AS. The existence of an enabled path in the PH level between each pair of objects in this subpath is then given by the LC1, corresponding to a PH-path in the form $sp_1 \langle p \rightsquigarrow v_j \rangle$. CA1 asserts that there is a PH-level connection corresponding to (c_1, c_2), said (v_j, v_{j+1}). Thus, for the path $\langle c_2 \rightsquigarrow T[o] \rangle$ the induction hypothesis can be applied, since it now spans $k-1$ ASs, achieving that there is a corresponding enabled PH path $sp_2 \langle v_{j+1} \rightsquigarrow o \rangle$. From the concatenation of sp_1 and sp_2, this results in an enabled PH-path $\langle p \rightsquigarrow o \rangle$, and CA2 thus implies the access from p to o is in this case possible. Q.E.D.

Therefore, the theorem VT1 results in that the validation criteria are satisfied in the abstraction refinement from the DAS to the PH level, when the local conditions and assumptions hold. Since the compliance of the DAS with the service-oriented view of the SR level was shown to be assured by the appliance of the global conditions and axioms (see sec. 5.2), thus we prove the seamless validity of the refinement from the SR level to the PH level according to the criteria of consistency and completeness.

Considering the evaluation of the conditions described in the previous sections, one can percieve an additional benefit that comes from the modeling technique presented in this paper: the possibility of splitting up the process of system analysis. Indeed, while the global conditions (GC1 and GC2) must be checked in the abstract view of the entire system (the DAS), in contrast, the conditions LC1 and LC2 only have a local scope. They can thus be performed independently on each AS and are restricted to consider only the elements inside the boundaries of a subsystem. Clearly, this analysis splitting contributes to a more scalable modeling, especially if we observe that the evaluation of the global conditions (that will be performed in a smaller, simplified view of the system) are more complex and computationally consuming than the local ones.

Further, the view of the system offered by the PH level is near enough to the implementation, such that it can be used to generate configuration files to the supported equipments and security mechanisms. This generation will be then based on the enabled paths that pass through each object, whose characteristics will be analyzed and converted into low-level configuration parameters. For the AS of Figure 3, for instance, an enabled path from the hosts "Workstation1" and "Workstation2" to the "Proxy Server" (through the processes "Netscape" and "Squid-Proxy" as well as through the corresponding protocol and interface objects) is analyzed and coverted into configuration parameters for the security mechanism "web proxy" in order to allow this access. As such, if we assume the correct generation of the configuration parameters from the PH level model, then the policies in the SR level will be correctly implemented; i.e. the actual system's mechanisms will be configured exactly in conformance with the high-level policy specification.

6. Conclusion

This paper has introduced a modeling technique that extends the Model-Based Management to enhance the handling of very large computing environments. We have briefly discussed the modeling improvements achieved by the Diagram of Abstract Subsystems and presented in detail the support this diagram offers to security policies.The supported policy types were analyzed and their relation with the other abstraction level entities was established.

We have also presented a model structure validation method that is made possible by our modeling technique. Since the modeling is based upon a policy-oriented view of the system, it enables—in addition to its main objective of automated policy refinement up to the generation of low-level configuration parameters for security mechanisms—the validation of the structural architecture of the system *vis-à-vis* its security policies, such that the criteria of *consistency* and *completeness* of the abstraction refinement can be proved. Moreover, we have shown that this validation can be performed in a modular fashion, which is more appropriate when dealing with large models.

Our approach is assisted by a software tool, which encloses a diagram editor (by means of which figs. 2 and 3 were drawn) and, once the modeling is complete, applies the checks described in Section 5 to the model. The tool has been used to perform a series of case studies and the results have presented enhancements both in modeling and analysis. The experience has also shown that, in practice, the modelled network systems are frequently not capable of enforcing the given high-level policies. In this case, the conditions we have presented (GC1, GC2, LC1 and LC2) can not be satisfied, and the tool offers indications to the necessary modifications on the system in order to make it congruous to the policies.

Future work includes integrating techniques for improving the visualization and navigation of the models.

7. Acknowledgments

We would like express gratitude to Helen Mary Murphy Peres Teixeira for reviewing, and to the German Academic Exchange Service (DAAD) for the scholarship funding.

References

[1] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1):34–64, February 1999.

[2] I. Lück, C. Schäfer, and H. Krumm. Model-based tool-assistance for packet-filter design. In E. Lupu M. Sloman, J. Lobo, editor, *Proc. IEEE Workshop Policy 2001: Policies for Distributed Systems and Networks*, number 1995 in Lecture Notes in Computer Science, pages 120–136, Heidelberg, 2001. Springer Verlag.

[3] I. Lück, S. Vögel, and H. Krumm. Model-based configuration of VPNs. In R. Stadtler and M. Ulema, editors, *Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002*, pages 589–602, Florence, Italy, 2002. IEEE.

[4] J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed system management. *IEEE JSAC Special Issue on Network Management*, 11(9), 11 1993.

[5] G. Rothmaier. Model-based security management: Abstract requirements, trusts areas, and configuration of security services. Master's thesis, University of Dortmund, Germany, 2001. in German.

[6] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[7] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2(4):333–360, 1994.

[8] M. Sloman and E. C. Lupu. Security and management policy specification. *IEEE Network, Special Issue on Policy-Based Networking*, 16(2):10–19, March/April 2002.

[9] R. Wies. Using a classification of management policies for policy specification and policy transformation. In Adarshpal S. Sethi, Yves Raynaud, and Fabienne Fure-Vincent, editors, *Integrated Network Management IV*, volume 4, pages 44–56, Santa Barbara, CA, 1995. Chapman & Hall.