On Scalability and Modularisation in the Modelling of 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
{jporto, paulo}@ic.unicamp.br

[2] FB Informatik, University of Dortmund, 44221 Dortmund, Germany
{joao.porto, heiko.krumm}@udo.edu

Abstract. As the use of computers and data communication technologies spreads, network security systems are becoming increasingly complex, due to the incorporation of a variety of mechanisms necessary to fulfil the protection requirements of the upcoming scenarios. The integrated design and management of different security technologies and mechanisms are thus of great interest. Especially in large-scale environments, the employment of security services and the design of their configurations shall be supported by a structured technique which separates the consideration of the system as a whole from the detailed design of subsystems. To accomplish this goal, this paper presents a scalable approach for the modelling of large security systems, relying on the concepts of policy-based management and model-based management.

1 Introduction

The widespread use of computers and data communication technologies, together with an ever-increasing Internet, requires the adoption of protection measures to control the risk of network-based attacks. In consonance with these protection needs, the technology utilised by security systems is growing in complexity. Thus, to the hardening of operating system configurations associated with the use of traditional firewalls [1,2], a series of mechanisms and services are incorporated like Virtual Private Networks (VPNs), end-to-end cryptographic associations (using, for instance, IPSec), authentication services (like Kerberos), authorisation services, and diverse monitoring, logging and auditing, as well as automated intrusion detection systems (IDS).

As those security services and mechanisms are increasingly employed—attaining thereby dazzlingly knotty scenarios—importance and costs of security management escalate. Initially, the management tasks are comprised of the installation and configuration of security services, followed then, during operation, by their monitoring, auditing, adaptation and reconfiguration. Proper abstraction, integration and tool support are thus key factors for easing the management tasks.

⋆ Scholarship funding by the German Academic Exchange Service (DAAD).

© Springer-Verlag Berlin Heidelberg 2005

Both policy hierarchies [3] and policy-based management [4] approaches can be profitably used in this context, since they aim at automating management tasks in complex systems. These two approaches work together as follows: management policy hierarchies can be built by initially taking a set of high-level policies and refining them through intermediate levels until reaching mechanically executable policies. Thus, policy-based management uses those relatively low-level policies with distributed management agents that will communicate with each other, interpreting and executing policies specifically assigned to corresponding management roles.

The Model-Based Management approach [5,6,7], in turn, supports the building of those policy hierarchies by means of interactive graphical design. It adopts concepts of object-oriented system design tools and employs a model of the system vertically structured into a set of layers. The objects and associations of a layer represent the system to be managed on a certain abstraction level.

A common problem of these approaches occurs when dealing with larger systems, since the model tends to lose much of its understandability, getting obscure due to the great number of components (as attested in [8]). A canonical way of addressing such problems is to use the principle of *divide and conquer*; i.e. the modularisation of a system into smaller segments would allow us to deal with each of them in detail separately, and to reason about the whole system through a more abstract view of the interaction of those parts.

In this paper, we apply this principle to achieve an approach based on the segmentation of a system into *Abstract Subsystems*. A *Diagram of Abstract Subsystems* constitutes thus a representation of the overall structure of the system in which the details are hidden and dealt with in the internal specification of each subsystem. This abstraction permits a decomposition of the processes of system analysis and design, thereby improving the comprehensibility and scalability of the model. Moreover, this diagram is policy-oriented and provides an interface between a service-oriented view and the depiction of the actual network mechanisms. This modelling technique is also assisted by a software tool, which consists of a graphical editor with additional functions for checking of model-dependent constraints and guiding the policy refinement through the model's hierarchical levels.

As the present work builds upon Model-Based Management, an introduction to the latter is given in the next section. Subsequently, the concept of *Abstract Subsystem* (AS) is presented (Sect. 3), to serve as a basis for the elaboration on the *Diagram of Abstract Subsystems* in Sect. 4 and on the modelling of ASs (Sect. 5). Section 6 discusses results from the application of our modelling technique to a realistic environment, and Sect. 7 presents the automatic model refinement. Lastly, we discuss related work in Sect. 8 and cast conclusions for this paper in Sect. 9.

2 Model-Based Management

The concept of Model-Based Management was initially proposed by Lück *et al.* in [5] and later applied to the configuration of several security mechanisms such as packet-filters [6] and VPNs [7]. This approach aims to support the 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.

The structure of the model is shown in Fig. 1 (reproduced from [8]), where three abstraction levels can be distinguished: *Roles & Objects* (RO), *Subjects & Resources* (SR), and *Processes & Hosts* (PH). Each level is a refinement of the superordinated level in the sense of a "policy hierarchy". The uppermost level represents the business-oriented view of the network whereas the lowest level is related to the 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) [9]. 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) 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*)—a service has references to all resources it is able to access; (d) the dependency of a service on other services (*ServiceDependency*); and lastly (e) *Resources* in the network. The *ServicePermission* class allows a subject to use a service to access a resource.

The SR level offers a transition from the business-oriented view, represented in RO level, to a more technical perspective, which is service-based. This is accomplished by using a service-oriented management approach to achieve a relatively abstract view of the management system, which is hence defined from the standpoint of the services that the system will provide. As such, the system's internal structure is not expressed in the SR level, but rather in the third level (PH) of the model (Fig. 1).

The lowest level (PH) is responsible for modelling the mechanisms that will be used to implement the security services defined in SR. Therefore, PH will

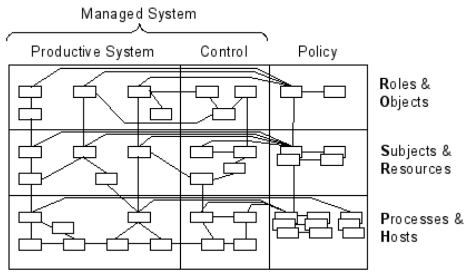

Fig. 1. Model Overview

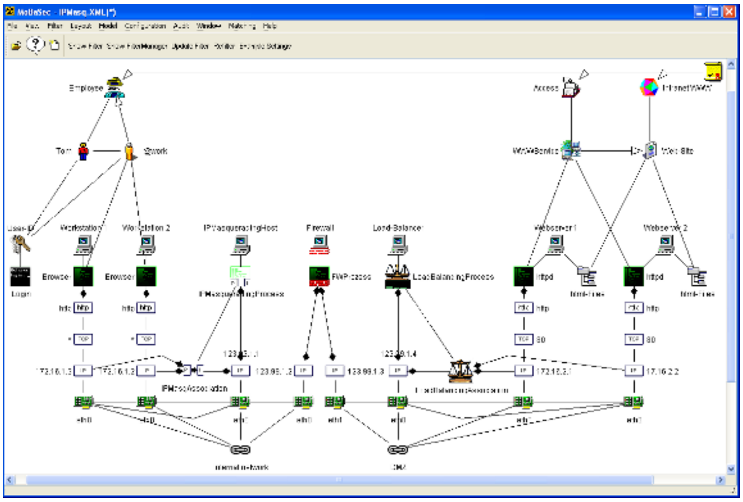

Fig. 2. Tool Interface

have even more classes than before, representing for instance the *Hosts*, with their respective network *Interfaces*, and the *Processes*, that perform communicative actions relying on *Sockets*. *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.

To support tool-assisted interactive configuration of mechanisms a graphical tool—whose interface is shown in Fig. 2—is supplied. This tool assists the user in the modelling of the system by means of a graphical editor with additional functions for the checking model-dependent constraints.

It can be noted from the previous discussion that the PH level—which shall depict the entire system, with its processes, hosts, network interfaces etc.—has its complexity quickly increased as the size of the modelled system grows. This fact is also illustrated in Fig. 2, which shows the model of a very simple scenario with only one *AccessPermission* at the uppermost level: the workers of a company shall be allowed to access the corporate web server. Despite the simplicity of this RO level, the model unfolds into a considerable number of objects at the lowest level (bottom of Fig. 2), in order to represent mechanisms like IP-masquerading, firewalls and load balancers.

Due to the simpleness of this example, the resulting model is still reasonable; however, it can be noted that models of larger real environments tend to become quite confusing. In order to overcome this problem we introduce in the next section the concept of *Abstract Subsystems*.

3 Concept of Abstract Subsystem (AS)

An *Abstract Subsystem* (AS) is an abstract view of a system segment; i.e. a simplified representation of a given group of system components. As such, an AS

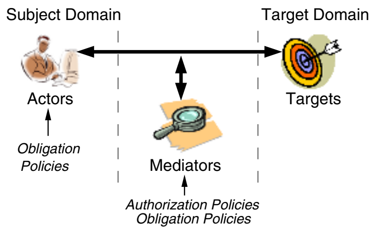

Fig. 3. Components of Abstract Subsystems

omits much of the detail that is contained inside of it, presenting instead only the relevant aspects for a global view of the system structure. These aspects are chosen based on a policy-oriented view of the system, which is depicted in Fig. 3.

In this scheme, three types of elements can be distinguished: *actors*, *mediators* and *targets*. The first type (*actors*) stands for groups of individuals in a system which have an *active* behaviour; i.e. they initiate communication and execute mandatory operations according to *obligation policies*.

The second element type encloses *Mediators*, which intermediate communications, receiving requests, inspecting traffic, filtering and/or transforming the data flow according to the *authorisation policies*. They can also perform mandatory operations based on *obligation policies*, such as registering information about data flows. The *Targets*, in turn, are *passive* elements; they contain relevant information, which is accessed by *actors*.

Using this scheme as a foundation, we can now redefine an *Abstract Subsystem* as a non-directed graph whose edges represent potential (bidirectional) communication between its nodes. These nodes can be either of one of the three types mentioned above (*actors*, *mediators* and *targets*) or *connectors*. This last type of component has been added to represent the interfaces of one AS with another; i.e. to allow information to flow from, and to, an AS.

4 Diagram of Abstract Subsystems (DAS)

Relying upon the concepts presented in the preceding section, we now introduce a new abstraction level into the modelling of security systems: the *Diagram of Abstract Subsystems* (DAS). This layer is located immediately below the service-oriented view of the system (SR level in Fig. 1) and above the PH layer, which depicts the actual network mechanisms. Its main objective is to describe the *overall structure* of the system to be managed in a modular fashion; i.e. to cast the system into its constituent blocks (ASs) and to indicate the connections between them. Since these blocks consist of a policy-oriented, abstract representation of the actual subsystem components (see Sect. 3), the DAS provides a concise and intelligible view of the system architecture—in a similar sense as the one proposed in [10]. Moreover, this diagram supports the reasoning about the structure of the system *vis-à-vis* the executable security policies, thus making

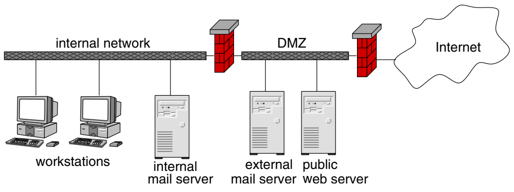

Fig. 4. Sample Network Scenario

explicit the distribution of the different participants of these policies over the system.

The DAS is formally a graph, comprised of ASs as nodes and edges which represent the possibility of bidirectional communication between two ASs, as shown at the bottom of Fig. 5. Thus, from a top-down perspective (i.e. from the SR level downwards) this diagram adds four kinds of information to the precedent model: (a) the segmentation of the users, services and resources into subsystems; (b) the structural connections amongst elements and subsystems; and therefore (c) the structural connections amongst the different participants of a policy (actors, mediators and targets); and lastly (d) mediators that are not directly related to SR level services but take part in the communication and filter or transform its data (e.g. firewalls).

In order to make clear the use of the DAS, in the next section we describe the systematic mapping from a service-oriented view of a system to its representation through abstract subsystems, and from this to the modelling of the actual mechanisms. Each step of this mapping is exemplified by means of a test scenario.

5 Modelling Abstract Subsystems

The scenario that will be used here relies on a typical network environment, as illustrated in Fig. 4. To this scenario the following network security policies are applied: 1) the users are allowed to browse the WWW from the internal workstations; 2) the users may send e-mails to the Internet from the workstations; 3) mail from the Internet shall be permitted to get to the external mail server, which in turn may forward it to the internal mail server; 4) external users using the Internet may access the company's public web server; and 5) users are allowed to fetch e-mails from the internal mail server to the workstations.

The modelling of these policies according to the principles referred in Sect. 2 produces the first two levels of Fig. 5. The basic objects are: the roles "Employee" and "Anonymous Internet User", and the *Objects* "Internal e-mail", "Website", "Internet e-mail" and "Internet WWW". These objects are associated with *AccesModes* by means of five *AccessPermissions* (at the top, on the right of Fig. 5), each corresponding to one of the above enumerated policies, which will henceforth be referred to as AP1 to AP5. Thus, for instance, the *AccessPermission* "allow Internet surfing" models the policy statement (1), associating the role

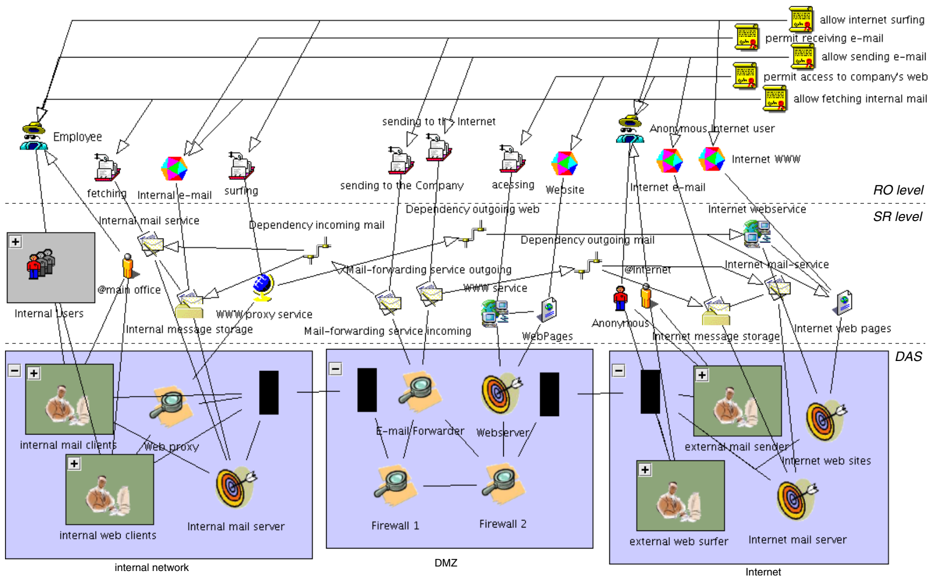

Fig. 5. Three-layered Model

"Employee" to "surfing" and "Internet WWW". The other policy statements are analogously modelled by the remaining *AccessPermissions*.

The mapping from the SR level to the *Diagram of Abstract Subsystems* (DAS) is then executed in three steps: (i) the modularisation of the system in conformance with the respective network scenario—i.e. the segmentation into *Abstract Subsystems* (ASs); (ii) the mapping of the elements of the SR level (users, services, resources) to components inside each AS; and (iii) the establishment of structural connections in the DAS, reflecting the associations between elements inside an AS and those between ASs, which are performed by means of *Connector* objects (Sect. 3). Subsequently, each AS can be expanded independently in order to achieve at the PH level a detailed representation of its mechanisms, and thereby enabling the generation of the corresponding configuration files. These steps will be described in turn in the following sections.

5.1 Segmentation into ASs

The subdivision of the system into ASs shall be guided by the structural blocks of the analysed environment. The abstract components of a DAS are thus aggregated according to the groups of mechanisms that already exist in the real system, such as departments, workplaces and functions.

An important criterium to be considered is the semantic unity of an AS; i.e. the group of mechanisms enclosed in an AS must have a common property that is clearly distinguishable. As such, this property assures the *cohesion* of the AS, so that it thereby represents a logical grouping identifiable in the real environment (like the ones previously mentioned), instead of only consisting of a mere agglomeration of heterogeneous elements.

On the other hand, the modularisation criterium of *coupling* (also a classical measure in the modularisation techniques of software engineering) should be taken into account in this context as well. The more the elements enclosed in the detailed view of an AS are related exclusively to elements of the same AS (and hence more independent from elements of other ASs), the more concise will be the abstract representation offered by the DAS, since those internal connections will be hidden. In this manner, a lower coupling between ASs improve the scalability of the DAS.

Analysing the scenario illustrated in Fig. 4, the existence of a structural subdivision in three segments is clear, namely: the internal network, the demilitarised zone (DMZ) and the external network (the Internet). Therefore, the DAS for this example has an AS to each one of these segments.

5.2 Mapping of Actors

An *actor* will be basically created for each group of hosts/processes (inside a given AS) which originates communication in order to act in conformance with a policy—which at the SR level is called service permission. In this manner, the actor corresponds to the *subject domain* of this permission, or, more precisely, to the part of the domain that is located in a given AS.

Nevertheless, actors can be shared by a number of different service permissions, as long as they have the subject domain comprehended by the actor in common. This contributes to a more compact representation, thus improving the scalability of the model.

In the SR level, the subject domain of service permissions is represented by *User* and *SubjectType* objects; thus each *Actor* will be connected to one or more pairs of these types of objects. In the framework of model-based management, however, service permissions are not directly modelled by the system designer but rather are automatically generated from *AccessPermission* objects located in the uppermost (RO) level. For this reason, the determination of the system's actors must start from an *AccessPermission*; thus taking the *Role* that is associated with it and identifying its corresponding *User* and *SubjectType* objects (in the SR level). Subsequently, one can create an *Actor* object that will contemplate the relevant *AccessPermission* and, consequently, is also related to the service permission that will be generated from it.

Considering our test scenario, an *Actor* object "internal web clients" is created in the AS "internal network" for the first policy—which is modelled by the first *AccessPermission* in Fig. 5, namely "allow Internet surfing" (AP1). Similarly, for the *AccessPermission* AP3 ("allow sending e-mail") the *Actor* "internal mail clients" is created in the same AS, whereas, in the AS "Internet", the actors "external mail sender" and "external web surfer" correspond respectively to the objects "permit receiving e-mail" (AP2) and "permit access to own web" (AP4). The *AccessPermission* "allow fetching e-mail" (AP5) can be covered by the previously created *Actor* "internal mail clients", since its subject domain is the same as that of AP3.

5.3 Mapping of Mediators

As regards to *Mediators*, two types can be distinguished. *Mediators* of the first type are a refinement from services which perform the "middleman functions" described in Sect. 3, for instance, proxies and mail forwarders. Therefore, they are achieved by means of a straightforward mapping from those services of the SR level, positioning them in the appropriate AS. Indeed, a service can be covered by a number of *Mediators*, each one residing in a different AS. On the other hand, a given *Mediator* object can map more than one service.

In our sample scenario, the "E-mail-Forwarder" *Mediator*, in the "DMZ" AS, stands for both services of handling incoming and outgoing e-mails. As for the "internal network", the "Web proxy" *Mediator* maps the "WWWProxyService".

The second type of *Mediators* consists of technical mechanisms that are not modelled in the SR level but are required in order to control the communication; i.e. they transform and/or filter it according to authorisation policies (like packet filters, IP-masquerading), or inspect the data according to obligation policies (e.g. IDS, event monitors). In this manner, the system designer shall create this type of mediators whenever these functionalities are required; i.e. a *Mediator* object will then appear wherever a security mechanism like the previously mentioned ones is to be placed in the respective actual network environment.

Examples of the second type of *Mediators* are illustrated in Fig. 5 by the objects "Firewall 1" e "Firewall 2". They have been introduced into the AS "DMZ", precisely in the place where the firewalls are found in the scenario of Fig. 4.

5.4 Mapping of Targets

Targets are obtained by a quite direct mapping from the pairs of *Service* and *Resource* objects (in the SR level) which encompass a target domain of a service permission, or a part of this domain that is placed in a given AS. In this way, each *Target* object must be connected to at least one pair of *Service* and *Resource* in the SR level, but it can also be shared by different service permissions; in the latter case, relations with other such pairs would be also present—this sharing also contributes to the conciseness of the model. Conversely, each pair of *Service* and *Resource* can be mapped to a number of *Targets*, each one located in different ASs—similar to the case of *Mediators*.

Similar to the actors, the target identification must start by considering the *AccessPermissions* in the RO level. Here, nonetheless, it is the *Object* related to a certain *AccessPermission* that is considered at first in order to establish then the corresponding *Resource* (SR level) and the *Service* which provides access to it. Finally, we create a *Target* to map this pair of objects.

When applying this method to our test scenario, then for the policy AP1 (Sect. 5) the *Target* object "Internet web sites" is created to refine the pair of *Service* and *Resource* of "Internet webservice" and "Internet web pages", since the latter is related to the *Object* "Internet WWW" (at the level RO) of AP1. It is worthwhile to note that, in this case, the *Service* refined from the *AccessMode* "surfing" of AP1 , namely "WWW proxy service", is different from the one previously used as target; this only happens when there is a *ServiceDependency* between these two *Services*. Indeed, the "WWW proxy service" cannot provide access to the *Resource* "Internet web pages" by itself, but relies on the "Internet web service" to do it. This dependency is modelled by the object "Dependency outgoing web" in the SR level (at the centre of Fig. 5).

Proceeding in the same manner, the targets "internal mail server" ("internal network"), "Internet mail server" ("Internet") and "Web server" ("DMZ") map respectively the policies modelled by AP2, AP3 and AP4. As for AP5, the *Target* "internal mail server" can be shared with AP2; as such, there is no need to create another object.

5.5 Establishment of Structural Connections

In order to complete the model that has been formed thus far by the application of the procedures of the latter sections, one needs to introduce the associations between objects of the DAS; i.e. formally speaking, to add the edges of the graph. Such associations have a different meaning compared to that of the associations between an element in the SR level and an object of the DAS. While the latter represent abstraction refinements—in the sense of relating levels of a policy hierarchy—the former represent structural connections; i.e. the possibility of communication in the actual system. Despite this, only the connections that are relevant to the abstract view of the system shall be depicted here; these are namely the associations that interconnect the different participants of executable policies: actors, mediators and targets.

Once again, the establishment of these connections starts at an *AccessPermission*. Each of the objects (in the DAS) that correspond to this permission is identified and then associations are created in order to construct paths between the respective actors and targets, traversing the necessary mediators. Whenever one of these paths enters or leaves an AS, *Connector* objects are inserted at this point, representing the communication interfaces of the AS. Thus, the number of *Connectors* in an AS corresponds to the number of available physical interfaces of the actual system. Proceeding in this manner with our test scenario, the *Connector* objects (rectangles) and connection edges (lines) shown in Fig. 5 are obtained.

5.6 Expansion of ASs

Starting from the model that has been produced thus far (Fig. 5), the next stage in the model development is to expand each of the ASs separately. This means that, for each AS, the mechanisms inside it shall be modelled according to the usual procedure described in [8], resulting in a detailed representation of these mechanisms; i.e. the PH level. Afterwards, the associations between the PH level components with the objects in the AS have to be drawn, thus establishing a relation of abstraction refinement.

Each *Actor* object of an AS must be then related to its corresponding *Process*- and *UserID*-typed components in the PH level, such that the *Actor* performs the association of these components with *SubjectType* and *User* objects in the

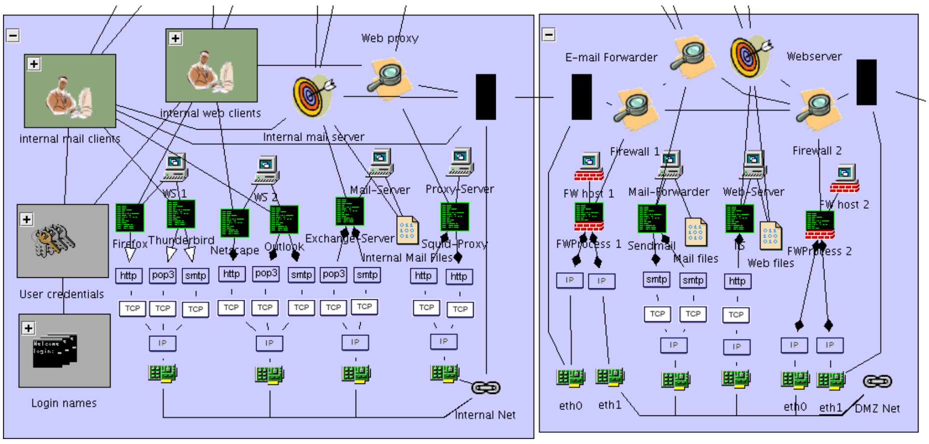

Fig. 6. Expanded ASs

SR level. As noted in Sect. 5.2, an *Actor* may be used for more than one pair of *SubjectType* and *User*, thereby corresponding to several service permissions. Hence, to avoid the burden of depicting all of the single associations amongst the objects (of type *User*, *SubjectType*, *UserID* and *Process*) connected to each *Actor*, a table of 4-tuples containing these associations shall be used to store them.

The *Mediator* objects of an AS, in turn, are simply related to one or more *Process*-typed components which implement the corresponding functionalities. With regards to *Targets*, each of them is related to one or more pairs of *Processes* and *Objects* that provide the corresponding services. Therefore, a *Process* in the PH level is related via a *Target* to a *Service* in the SR level, whereas an *Object* is related in a similar fashion to a *Resource*.

Figure 6 presents the PH level model for the ASs "internal network" and "dmz" in our example (see Fig. 5). In Fig. 6, the relation from actors, mediators and targets in the AS (at the top) to objects in the PH level (bottom)—achieved merely by following the principles previously exposed in this section—is graphically indicated by edges. For instance, the *Actors* "internal mail clients" and "internal web clients" in the "internal network" (on the left) are related to PH-level objects that represent the corresponding processes that run in two different workstations, as well the user credentials and login names of the users that may take advantage of these processes. On the other hand, the *Mediator* "E-mail Forwarder" and the *Target* "Webserver" in the AS "DMZ" (on the right) are correspondingly mapped to processes and resources that implement them. These PH objects directly assigned to AS entities are in turn related to a series of other PH objects in order to provide the model with detailed information about the communication, such as the protocol stack and the network interfaces used (see bottom of Fig. 6). In this manner, the correspondence between the abstract view of the system (AS) and its actual mechanisms (PH level) is established.

6 Application Case and Results

In this section we report about the application of the modelling technique presented in the previous sections to a realistic large-scale network environment. This environment (inspired from the one in [8]) consists of an extension of the simple test scenario of Sect. 5, in order to address an enterprise network, composed of a main office and branch office, that is connected to the Internet.

The employees are allowed to use the computers in the main and branch offices in order to surf on the Internet and to access their internal e-mail accounts, as well as to send e-mails to internal and external addresses. In addition to that, they may also retrieve their e-mails from home. As for the ongoing communication from the Internet, the security system must enable any external user to access the corporate's web site and to send e-mails to the internal e-mail accounts. Our main goal is then to design the configuration for the security mechanisms that are required to enable and control web-surfing and e-mail facilities for the company's office employees, namely: three firewalls, three VPN gateways, and a web proxy.

Figure 7 presents the DAS obtained for this environment. It is composed by five ASs, representing the logical network segments of the described scenario: "internal network", "dmz", "Internet", "branch office's network" and "remote access point". The *Actors*, *Targets* and *Mediators* of each of these AS and their interconnections through *Connectors* and structural connections are also depicted in Fig. 7. Pictures of the model's overview for the application case and of the detailed PH-level models for the ASs "internal network" and "dmz" are given in the Appendix A.

Analysing the model for this realistic application case, one can clearly perceive the advantages brought forth by the DAS. Altough the detailed information of the PH level encompasses more than 500 objects—representing, for instance, 8 hosts and credentials for 30 users in the internal network, 8 hosts

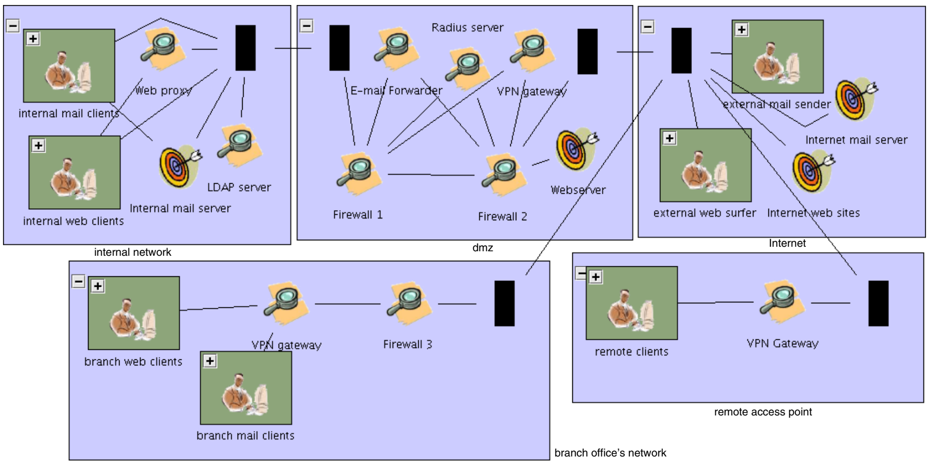

Fig. 7. DAS of a complex environment

in the branch office's network and a cluster of 5 web servers in the "dmz" (see Appendix A)—the abstract representation of the DAS (Fig. 7) consists of only 32 objects alltogether. Therefore, it can be noticed that the modelling through abstract subsystems offers concrete advantages in the conciseness and understandability of the model.

Furthermore, since the environment in this application case is an extension of the test scenario used in Sect. 5, it is possible to compare them, in order to identify the growth behaviour of a DAS. The number of PH objects (which depict the mechanisms of the actual system, and thus reflect the growth of the system itself) increased from 95 in the simple test scenario to 540 in the realistic application case (i.e. a growth of almost 470%). The number of DAS elements, in contrast, rose from 19 (Fig. 5) to 32 (Fig. 7)—i.e. a growth of only less than 69%. We thus conclude that the size of a DAS does not increase in the same pace as the number of elements in the PH level (and thus as the system's mechanisms), but rather the DAS's growth is much slower. This makes clear the scalability gain afforded by the DAS in the support of large models.

Since the number of elements both in the DAS and in the PH levels heavily depend on intrinsic characteristics of the environment modelled (such as the entities to be modelled and the possibility of subdividing and grouping them), an unrestricted generalisation of these quantitative results is not possible. Nevertheless, in qualitative terms, similar gains can be expected in the modelling of other large-scale networked environments; for they are similar to the typical scenarios presented here.

7 Tool Support and Automated Refinement

To support our modelling, a software tool is provided. This tool encloses a diagram editor (by means of which Fig. 5, Fig. 6 and Fig. 7 were produced) that allows the inputing of model objects and their relationships, as well as the assignment of properties to them. In this manner, each step of the modelling explained in the previous sections is assisted by the tool, which verifies the compliance of the model with structural constraints in the moment particular objects are inputed. Once the modelling is complete, a series of checks are performed to assure the consistency of the model as a whole.

Though the fully automated derivation of low-level, executable policies from a set of abstract specifications is, in the general case, not practical [11,12], our modelling 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. Thus, the analysis of the system's objects, relationships and policies of an abstraction level enables the generation of lower level policies, based on the system's model in the lower level and on the relations between the entities of the two layers[1].

[1] An extensive elaboration on the policy refinement process and on the consistency checks that are described in this section is beyond the scope of this paper and can be found in [13].

In this process, the support tool firstly derives each one of the given *AccessPermissions* (in the uppermost level RO) into one or more *ServicePermissions* in the SR level (see Sect. 2). Afterwards, a set of *ATPathPermission* (ATPP) objects are generated from the *ServicePermissions.* Each ATPP is a path in the graph (DAS) that represents the permission for an *Actor* to reach a certain *Target* passing through the required *Mediator* and *Connector* objects. The refinement advances with the automatic generation of *ProtocolPermissions* from the ATPPs, using the detailed information contained in the PH level in order to achieve the security policies for this level. Each *ProtocolPermission* is then related to a set of objects, denoting that an initiating process—to which a user credential can be assigned—may communicate, via its co-located protocol entity and a remote protocol, with a serving process in order to access a certain physical resource.

Throughout the above refinement process, a series of conditions are verified against the model, in order to check the consistency of the different abstraction levels and the feasibility of enforcing the high-level policies defined by the administrator. Our experience also shows that, in practice, the modelled network systems are frequently not capable of enforcing the given high-level policies. In this case, the consistency checks cannot be satisfied, and the tool offers indications to the necessary modifications on the system in order to make it congruous to the policies.

Finally, for the last step of model-based security service configuration, a series of back-end modules are executed, where each module corresponds to a special security service product (e.g. Kerberos, FreeS/WAN, Linux IP tables etc.). These back-end functions evaluate the *ProtocolPermissions* and the PH model in order to generate the adequate configuration files for each of the security service products. Further details can be found in [6,7,8].

8 Related Work

There are a considerable number of approaches to policy specification both for security management and policy-driven network management purposes (see [11] 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 graphical tool Firmato [14] seems to be the closest available approach to ours, since it supports the interactive policy design by means of policy diagrams and automatically derives the corresponding configuration parameters for mechanisms such as routers, switches, and packet filters. However, since the abstraction levels of graphical policy definitions and configuration parameters are relatively near to each other, its support is restricted to an abstraction level that is close to the system mechanisms. In this respect, the Power prototype [15] has a broader scope, aiming to support the building of policy hierarchies by means of a tool-assisted policy refinement. Nevertheless, Power does not allow a free graphical definition of policies, relying instead on pre-defined templates and wizard engines. Furthermore, neither *Power* nor *Firmato* are concerned with scalability issues; this fact is reflected by the absence of a modular system's representation in these approaches.

9 Conclusion

This paper has presented a modelling technique for the management of security systems. The modelling achieves scalability by the segmentation of the system in *Abstract Subsystems*, which enables the processes of model development and analysis to be performed in a modular fashion.

The systematic mapping from a service-oriented system view to a *Diagram of Abstract Subsystems* was covered in detail, encompassing the choice of elements to be represented in the abstract view, as well as the correspondence of these elements to the actual mechanisms of the system. A realistic case study was presented and the results achieved through our modelling were discussed. We have concluded that concrete gains in the understandability and scalability of the modelling of large-scale systems can be expected from the employment of our technique. Furthermore, the tool-assisted modelling and automated policy refinement supported by our prototype tool were also briefly described.

Future work could include improving the representation of policies at the lower levels of the model, in order to ease their handling.

Acknowledgments. We would like express gratitude to Helen Mary Murphy Peres Teixeira for reviewing.

References

1. Cheswick, W.R., Bellovin, S.M., Rubin, A.D.: Firewalls and Internet Security: Repelling the Wily Hacker. 2nd edn. Addison-Wesley (2003)
2. Zwicky, E.D., Cooper, S., Chapman, D.B.: Building Internet Firewalls. 2nd edn. O'Reilly and Associates, Sebastopol, CA (2000)
3. Moffett, J.D., Sloman, M.S.: Policy hierarchies for distributed system management. IEEE JSAC Special Issue on Network Management **11** (1993)
4. Sloman, M.: Policy driven management for distributed systems. Journal of Network and Systems Management **2** (1994) 333–360
5. Lück, I., Schönbach, M., Mester, A., Krumm, H.: Derivation of backup service management applications from service and system models. In R. Stadler, B.S., ed.: Active Technologies for Network and Service Management, Proc. DSOM'99. Number 1700 in Lecture Notes in Computer Science, Heidelberg, Springer Verlag (1999) 243–255
6. Lück, I., Schäfer, C., Krumm, H.: Model-based tool-assistance for packet-filter design. In M. Sloman, J. Lobo, E.L., ed.: Proc. IEEE Workshop Policy 2001: Policies for Distributed Systems and Networks. Number 1995 in Lecture Notes in Computer Science, Heidelberg, Springer Verlag (2001) 120–136
7. Lück, I., Vögel, S., Krumm, H.: Model-based configuration of VPNs. In Stadtler, R., Ulema, M., eds.: Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002, Florence, Italy, IEEE (2002) 589–602

8. Geist, G.: Model-based management of security services: Integrated enforcement of policies in company networks. Master's thesis, University of Dortmund, Germany (2003) in German.
9. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer **29** (1996) 38–47
10. Porto de Albuquerque, J., de Geus, P.L.: A framework for network security system design. WSEAS Transactions on Systems **2** (2003) 139–144
11. Sloman, M., Lupu, E.C.: Security and management policy specification. IEEE Network, Special Issue on Policy-Based Networking **16** (2002) 10–19
12. Wies, R.: Using a classification of management policies for policy specification and policy transformation. In Sethi, A.S., Raynaud, Y., Fure-Vincent, F., eds.: Integrated Network Management IV. Volume 4., Santa Barbara, CA, Chapman & Hall (1995) 44–56
13. Porto de Albuquerque, J., Krumm, H., de Geus, P.L.: Policy modeling and refinement for network security systems. In: Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden (2005) 24–33
14. Bartal, Y., Mayer, A.J., Nissim, K., Wool, A.: Firmato: A novel firewall management toolkit. ACM Transactions on Computer Systems **22** (2004) 381–420
15. Mont, M., Baldwin, A., Goh, C.: POWER prototype: Towards integrated policy-based management. In Hong, J., Weihmayer, R., eds.: Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS2000), Hawaii, USA (2000) 789–802

A Models of the Application Case

We present here some of the models obtained for the application case analysed in Sect. 6. The growth in the complexity of the PH level is made clear from the comparison of the models of the AS “internal network” in the realistic application case (Fig. 8) with that in the test scenario (left hand of Fig. 6), and similarly for the AS “dmz” (compare Fig. 9 with the right hand of Fig. 6).

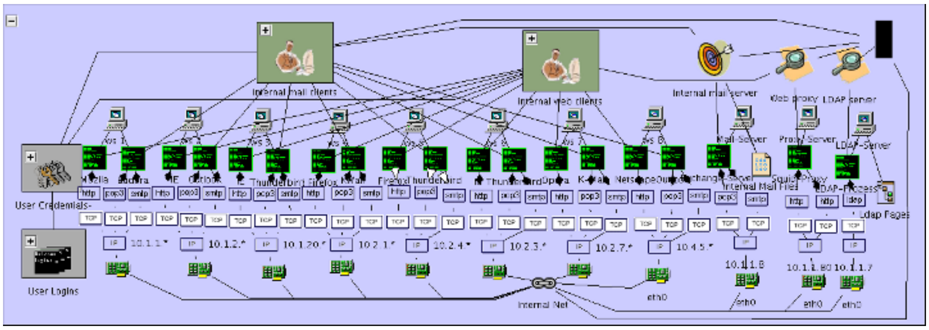

Fig. 8. PH-model of the AS “internal network”

In contrast, the superior levels of the modelling show a slower growth behaviour, and hence more scalability. Figure 10 presents the three-layered model for the application case (compare with Fig. 5).

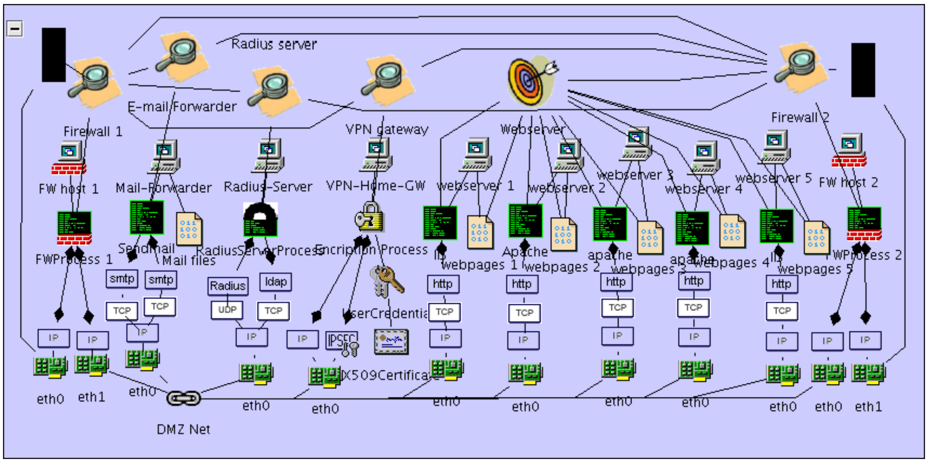

Fig. 9. PH-model of the AS “dmz”

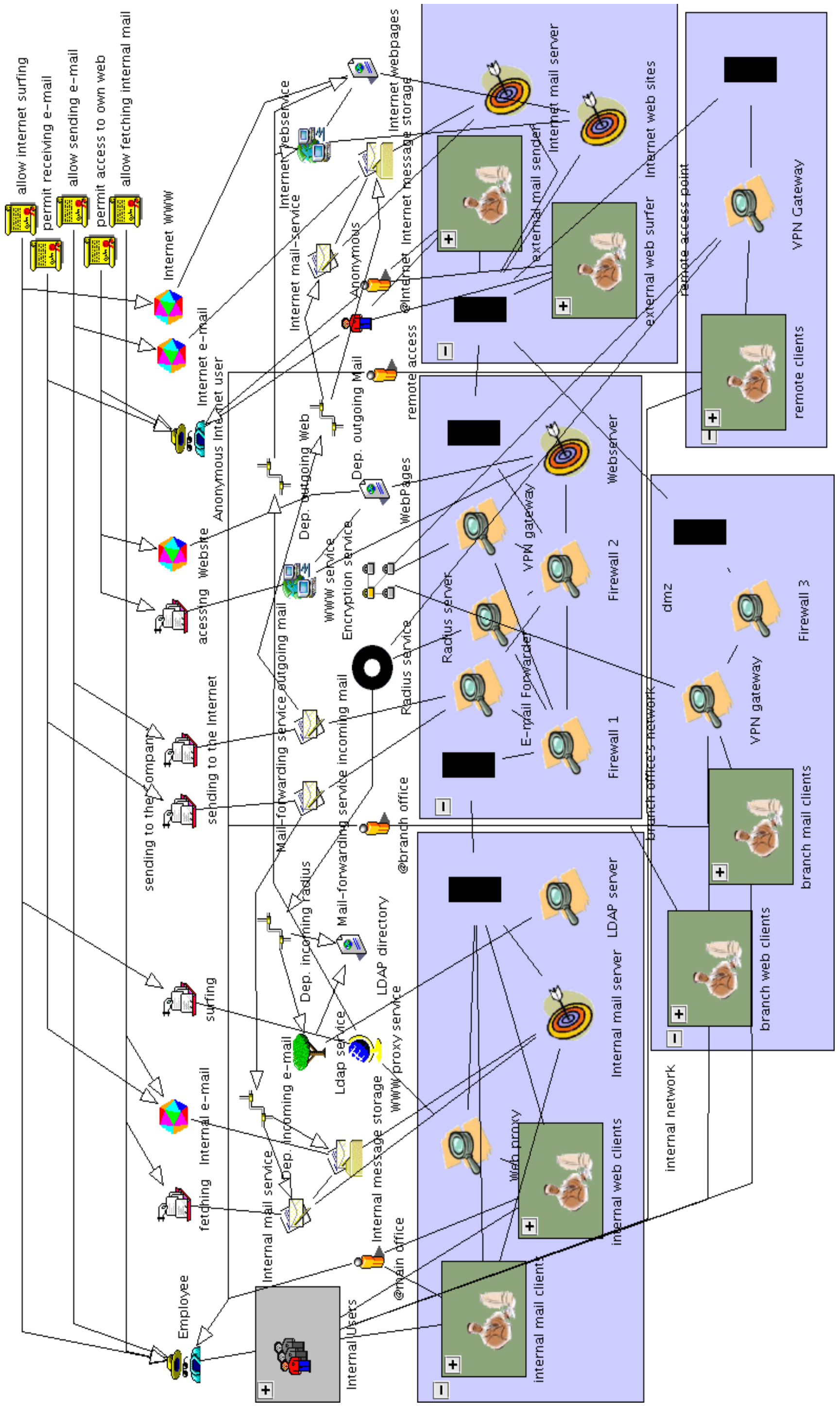

Fig. 10. Three-layered model of the Application Case