Security Service Adaptation for Embedded Service Systems in Changing Environments

Stefan Illner, Andre Pohl and Heiko Krumm

FB Informatik, Universität Dortmund, D-44221 Dortmund, Germany
e-mail: (stefan.illner, andre.pohl, heiko.krumm)@udo.edu

***Abstract*—Distributed embedded applications increasingly operate in changing environments where the application security depends on the type and properties of the currently used communication services and employed devices. While vulnerabilities, threats, and available security function processing power are changing, the applications, however, should automatically adapt to the varying conditions in order to maintain the necessary security without endeavor of users. We report on the security management subproject of the *SIRENA* project where we apply a special combination of *policy-based management* with *model-based management* in order to support fully automated security management functions at runtime as well as tool-assisted security requirement definition and system design. Within an application model, the definition of the application's high-level security policy is of special importance. It represents the abstract security requirements and forms the starting point for the automated derivation of suitable security subsystem configurations which enforce the policy under changing environment conditions. The abstract policy representation relies on the *Generalized Role Based Access Control model (GRBAC)*.**

***Index Terms* — embedded service system security, automated security service configuration, GRBAC model, model-based management.**

I. Introduction

The automotive, industrial, home, and telecommunication domains perceive an increasing demand for embedded real-time applications which rely on cooperating networked devices. A plethora of devices is arising covering small and specialized devices with restricted computing power as well as well-equipped servers and workstations. Stationary and mobile devices cooperate in dynamically varying associations and utilize a wide variety of supporting services. Mobility, changing tasks, as well as fluctuating supportive service costs and availabilities demand that the frequently needed adaptations of application configurations are accomplished by automated application management functions.

Usually there are communication services, supportive services, and devices in use which may be exposed to attacks threatening application objects and functions. Consequently IT-security is a major concern. The applications have to employ security services like authentication, authorization, access control, non-repudiation, key management, and encryption which may introduce substantial memory, processing and bandwidth load. When the communication route of sensitive information suddenly involves insecure media, strong encryption has to be employed which might not yet be used before in order to avoid unnecessary overhead. When, moreover, one endpoint of that communication is located on a small device which is not able to execute strong encryption algorithms, a further device has to be engaged which acts as proxy and provides the endpoint-near encryption [13]. That way the security services used by an application shall dynamically be reconfigured during runtime adapting their configuration perpetually to the current needs and facilities. The planning and design of suitable security service configurations, and – at runtime – their initial establishment as well as their control and adaptation comprise the tasks of the so-called security management.

We study the security management of distributed embedded applications in the context of the project "*Service Infrastructure for Real time Embedded Networked Applications (SIRENA)*" [11], which transposes the notions of service-orientation and service-structuring from the general setting of web-services to distributed real-time applications. Here, service orientation particularly supports the design of adaptable application architectures. *SIRENA* defines a framework providing suitable application services and supportive services in connection with corresponding interface definitions and implementation components.

We aim at a high automation of the security management tasks applying a combination of the approaches of policy-based management [12], policy hierarchies [9], and model-based management [7]. The *runtime tasks* are performed by an automated security management application subsystem which cooperates with the general automated configuration management system. The *design time tasks* are supported by special tool-assistance. An interactive graphical *modeling tool* supports the definition of an object-oriented application model which represents the application's layered implementation configuration as well as its high level security policy. The high level security policy defines the abstract security requirements of the application. After interactive model definition, the tool is able to translate the abstract policy automatically into policy enforcing security service configurations as follows. It adds corresponding security service representations and management system components to the model. Moreover it computes the low-level policies which control the automated management functions. Finally backend functions of the tool generate configuration description files which support the automated employment of the application together with its integrated security and management subsystems. The *high level policy* is defined

in terms of the *Generalized Role Based Access Control Model (GRBAC)* [10] which is an extension of the *Role Based Access Control Model (RBAC)* [3].

In the sequel we give an outline of and an example for *SIRENA* applications and enter into their security requirements. After describing *GRBAC* we explain its utilization for the definition of security requirements. Moreover we present the principles of the meta-model the tool-functions rely on.

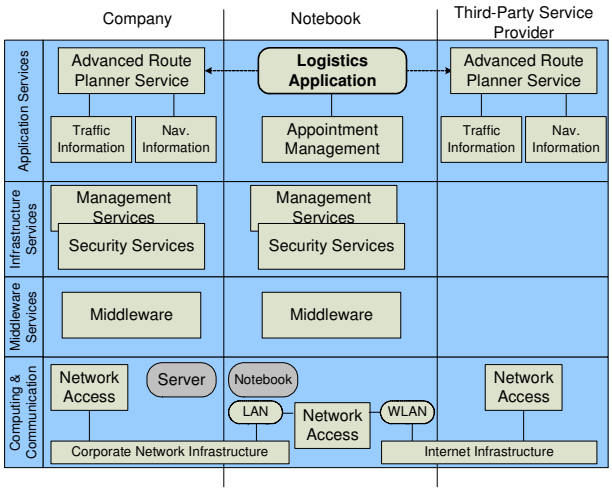

Fig. 1: Proposed *SIRENA* Service Architecture

II. Embedded Service System Applications

Structure and properties of embedded applications shall be clarified by means of an example which is depicted in Fig. 1. The structure is stamped by four layers where the topmost layer contains the so called *application services*. These services implement functionalities which can be invoked and used by user-space applications, e.g. appointment planning or routing services. The next layer of the architecture contains common management and security services. These *infrastructure services* are responsible for managing the components of the lower layers and thus also for the automatic adaptation to changing environmental conditions. The *middleware services* layer provides abstract connection and communication facilities. It masks different physical connection techniques to provide a single abstract interface to the upper layers. The lowermost *computing and communication* layer contains the physical hosts and their operating system services which provide the base functionalities to enable networked communication between *SIRENA* services. A particularity of the *SIRENA* environment is that the lower three layers may change during an application's runtime regarding connectivity and security issues. The application itself may remain unchanged and even may not be aware of the changes in the lower layers.

In the application layer of Fig. 1 a simple application scenario is modeled. Imagine a company which employs several people acting as field staff personnel each equipped with a notebook running a logistics application. This application manages the planning of traveling routes in dependency of the appointments for an employee:

1. The field staff member is at his office and starts the logistics application on the notebook. The logistics application uses the company's *Advanced Route Planner Service* to compute an appropriate route to fulfill all appointments.
2. The employee is on-the-road and due to a traffic jam or other adversities the planned route cannot be used anymore. He starts the logistics application to compute a new appropriate route. The logistics application locates an advanced route planner service and uses it for its computations.

From the application's point of view, these two scenarios differ only slightly. On the lower layers, however, substantial changes exist and have to be managed at runtime.

III. Security Requirements

In order to integrate security services into an application and to design their configurations, a *security analysis* is accomplished. It iteratively performs the following steps [1]. The analysis starts with identifying the components of which the system is built. All valuable components that have security needs are referred to as *assets*. For every asset a *required security level* has to be stated, with regard to security concerns like integrity, confidentiality, and availability. These levels depend on the value of the asset and on its security objectives. Hereafter the *threats* threatening the assets must be identified. Along with this, *vulnerabilities* that may be utilized by threats must be uncovered. The resulting system view serves as a starting point for assessing the resulting *risks*. The last step consists of planning countermeasures which reduce the risks to an acceptable degree.

The security analysis is based on the International Standard which is called *Common Criteria* (*CC*) [6]. *CC* contains rules for conducting a security analysis in a standardized way and provides a common analysis framework that makes the evaluation results comparable and thus wider acceptable. *CC* establishes a set of functional components as a standard way of expressing the functional requirements for *Targets of Evaluation* (*TOE*). Moreover it states security functions that can be utilized to protect the *TOE*s.

We apply the special object-oriented and model-based security analysis approach introduced in [5]. It supports the direct model representation of the notions of *CC*. A system is modeled by an object instance diagram using a graphical interface, and is analyzed using a set of graph transformation rules. The objects used in modeling are instances of classes describing the components and assets that exist in the real system. Further classes are introduced for modeling vulnerabilities, threats, risks, and countermeasures. Every threat has an associated likelihood, describing the probability that this threat may successfully attack an asset. It is associated to one or more assets. The risk an asset encounters depends on the required security level and on the threats it is facing. Countermeasures have associated levels of protection they provide for each security concern and are associated to the assets they shall protect.

The second part of *CC* defines the so called *security*

functional components, which can be composed to *Security Targets* describing the desired security properties of the analyzed system. The functional components express security requirements intended to counter threats, but they do not detail how to achieve this, just state the mechanism types. After having modeled a system and having performed the security analysis, the security components have to be identified which are necessary to reach the desired protection level. In the next step these components are structured into groups of similar functionality. *CC* guides the structuring by a subdivision of security functions into classes, families, components, and functional elements. All functional components that are needed for protection purposes have to be incorporated into a *SIRENA* application in form of security services.

In the *SIRENA* framework, security services realizing a broad variety of security functional components exist. Two examples shall clarify relevant properties. In the *Specification of secrets* (*FIA_SOS*) family, methods are summarized that deal with quality metrics of secrets, in checking them for existent secrets, and in applying them for the generation of newly created secrets. Depending on device capabilities, metrics can be set to generate secrets that provide only weak security, or secrets that provide high security, but need much computational power when generating them and using them in cryptographic algorithms. For encryption, signature generation, and further similar or related actions, functionality of the *Cryptographic operation* (*FCS_COP*) family is needed. During the service design phase, a set of cryptographic algorithms and corresponding key sizes must be specified.

The protection needs of an application can vary due to changing environments. The security analysis therefore has to prepare the runtime *adaptation* of the employed security services. At design time, it has to consider all possible environment scenarios and to propose a suitable *security service configuration* for each scenario. At runtime, the basic handling of the configurations – i.e., the detection of environment changes and the execution of configuration transitions – is performed by the *general application configuration management*. There, environment change events are created and trigger the runtime *security service management* which selects the appropriate security service configuration transitions and requests their execution. After execution of a requested transition by the general configuration management, the security service employments are adapted to the new situation. Since the underlying security analysis was performed at design time, complex runtime decisions are avoided and the method is also suitable for small devices.

CC aids in determining the necessary management functions by describing management requirements for each security function family. This provides a good starting point for identifying the relevant adaptation management issues. In particular, following requirements apply. The *Management of functions* (*FMT_MOF*) has a key role since it allows authorized users the management of functions. It provides methods for managing the users that are allowed further security function managing. The focus of *Management of security attributes* (*FMT_MSA*) are those functions that deal with security attributes, and ensure that the assigned values are reasonable. *Cryptographic key management* (*FCS_CKM*) supports all functionality required for handling cryptographic keys, including key generation, distribution, and destruction.

In the example outlined above, we assume two environments being present, one that is in place when the employee is in his office connected to the company LAN, and the other when he is on-the-road. For each of these environments, a special security service configuration is established. In the office, connection to the internet may happen without restrictions or security concerns, as the company's IT department uses firewall and virus protection products. Outside, WLAN-based internet access may be used, which is inherently insecure. Thus the security service configuration must ensure that the communication with servers of the company is encrypted using specified algorithms and the corresponding keys. The design time security service configuration management is responsible for selecting appropriate algorithms and keys, while the runtime management has to observe environment changes and perform the necessary transitions between configurations.

IV. GENERALIZED ROLE-BASED ACCESS CONTROL MODEL

The *Generalized Role-Based Access Control* (*GRBAC*) model is an extension of the well known *Role-Based Access Control* model ([4], [3]). In the following these two models are outlined briefly.

Role-Based Access Control (RBAC)

The *RBAC* approach uses roles instead of a per user access control policy to model access permissions. These roles are linked to the structure of an organization and carry specific access rights. Users or applications acting on behalf of a user are referred to as *subjects* in the *RBAC* terminology. Subjects are assigned a set of *authorized roles*, i.e. each subject can only act in a role that is defined in this set. In addition the *RBAC* model defines so called *objects* and *operations*. Objects are the entities protected by a *RBAC* system, e.g. services or files. Operations describe sequences of accesses of subjects acting in specific roles to *RBAC* objects. A formal definition of the *RBAC* model follows [10]:

Subject S	a user representation in the system
Role R	a categorization primitive for subjects
Object O	a system resource
Operation T	a sequence of one or more accesses to one or more objects
AR(S)	the authorized role set for subject *S*
AT(R)	the authorized operation set for Role *R*
exec(S, T)	*true* iff subject *S* is authorized to execute operation *T;* *true* iff $\exists$ role *R: R*$\in$ *AR(S), T*$\in$ *AT(R)*

Based on the active set of subject roles, the operation and the associated object a system based on the RBAC model decides whether or not the access should be granted.

Generalized Role-Based Access Control

GRBAC [10] extends the *RBAC* model outlined above by adding two more types of roles: *object-* and *environment-roles*. Object-roles are intended to model the type or internal state of a RBAC object. This can be a role that describes if an object is able to handle some kind of specific data, or when considering information or data as an object itself this type of role may describe the type of information provided. Environment-roles are used to model the environment of an application. This can be the time of day or the weight of the subject trying to access a specific object.

Due to the addition of two more roles, the decision process to determine whether or not an access should be granted is more complicated. This process is as follows:

1. It is checked, if there exists an object-role R_O of object O, and
2. an environment-role R_E which is active at the moment, and
3. a transaction/operation T which grants a subject-role R_S access to an object O in the role R_O while the role R_E of the environment is active.

V. GRBAC-BASED APPLICATION MODELING

Due to the inherently mobile character of many applications, an application's environment is subject to ongoing changes which have to be reflected by adaptations of the security service configuration in order to maintain the required security levels. The configuration design and the planning of adaptations need a precise but abstract definition of the security requirements and their dependencies to environment changes, as they are identified in the security analysis process, and we apply *GRBAC* for that purpose.

The abstract state of a system can be expressed by properties. The *GRBAC* model extends this view by allowing roles to reflect the states of a system by incorporating the notion of object- and environment-roles. Due to this, a system's state can be modeled as a set of active roles which are bound to different properties of the system. Changes in the system's state result in a different set of active roles. To model the state of the environment of *SIRENA* applications four types of roles have been identified:

- *Subject-Roles*: Each user (or an application which acts on a user's behalf) can act in an authorized subject-role.
- *Device-Roles*: A device is characterized by its computing power, resources and other properties.
- *Service-Roles*: The notion of a service is a central part of the *SIRENA* framework. This type of role thus specifies properties of a service, ranging from connection issues to trust relationships.
- *Environment-Role*: The notion of an environment-role is adopted from the GRBAC definition.

The security requirements of a *SIRENA* application are modeled using a vector which assigns security levels to security concerns like confidentiality, integrity, availability and accountability. Security levels are defined in an abstract manner using the values *low*, *medium*, *high* and *very high*. These different levels correspond to the risk of a successful attack to the corresponding security issue. The risk value is computed by correlating the probability of a successful attack with the required level of security associated with a given asset. This risk value directly defines the strength of security functions which should be used to secure and protect the asset. These security functions therefore have to be classified by their strength (and maybe also their cost) so that a certain risk level can be directly accompanied with a set of corresponding security functions.

Given the above elements of the model, namely roles describing the current system-state and security vectors reflecting the risk of a successful attack, a model of an application scenario can be created which provides a basis to derive an appropriate configuration of security services. These scenarios contain a description of the application's environmental state and a security requirements specification bound to this state. Using this model it is possible for an administrator or application developer to create an abstract configuration description without looking at the low level security service configurations.

To enable the adaptive and automatic reconfiguration of the security services regarding to the current application state, suitable security service configurations have to be established at runtime. Configuration management services

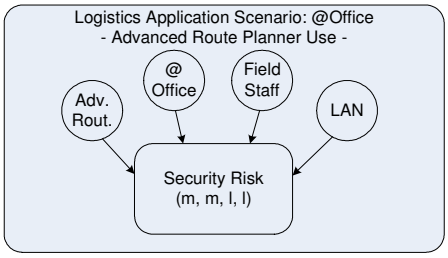

Fig. 2: Security Requirement Profile (@Office)

observe the current state of the application's environment and automatically perform the transition to the appropriate service configuration. The security management services are responsible for managing appropriate security functions and mechanisms which can be used to enforce a specific security level. By looking at these functional components the model is divided into three layers, each reflecting a different level of abstraction:

1. Roles & Security Requirements (modeled by the administrator or developer)
2. *SIRENA* Security Services (configured using the configuration created from the abstract model)
3. Security Functions & Mechanisms (used to enforce a specific level of security)

In the following a simple example of modeling application security requirement profiles is given, referring to the *SIRENA* application example given in section II.

In Fig. 2 a simple security requirement profile is modeled using the elements introduced above. The profile consists of four different roles describing the applications environment in an office use situation. The *Adv. Rout.* service-role represents the properties of a common service for advanced route planning. The *@Office* environment-role reflects the location of the application use. The group of subjects using the service in this scenario is modeled by the

Field Staff subject-role. The type of connection used by the *SIRENA* middleware services to provide a communication facility to the application is modeled through the *LAN* service-role. These rules are coupled with a *Security Risk* element. This element defines a *medium (m)* risk for confidentiality and integrity security issues and a *low (l)* risk for availability and accountability issues. This scenario may be typical for an office network where all resources are controlled by a single trusted party and the network infrastructure is secured through firewalls and switched subnets.

The second scenario depicted in Fig. 3 models the same application, used in an *On the Road* situation where a field staff member is en route using his car und carrying the notebook containing the logistics application with him. Due to the use of the *SIRENA* framework the application-code may not be aware of the changing environment and thus does not even recognize that the field staff member has left the office complex. The application simply tries to locate an advanced route planner service and uses this one exactly like the other one at the office. In contrast to the office scenario this time the service is provided by a 3rd party. In addition the network connection used to communicate with the service has changed from the relatively secure office network to a WLAN access point. These context changes are modeled using appropriate roles reflecting the environment, service and subject properties. As you can see, the security requirements also altered. Due to the insecure nature of WLAN networks and the fact, that the routing service is provided by a 3rd party, the confidentiality risk-level is raised to *very high (vh)*. Also the other levels for message integrity, availability and accountability are raised to the *high (h)* level.

For each usage scenario in different environments security requirement profiles have to be modeled to enable the

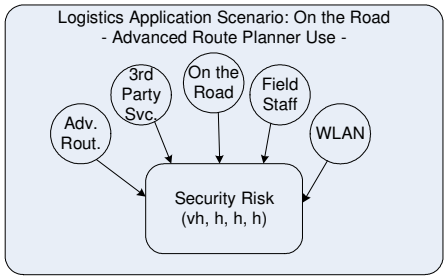

Fig. 3: Security Requirement Profile (On the Road)

adaptive automatic configuration of the *SIRENA* security services.

VI. MODEL-BASED MANAGEMENT

As outlined in the introduction there are design time and runtime security service management tasks. The design time tasks are supported by an interactive graphical modeling tool. The runtime tasks are performed by components providing general management (in particular configuration management), security services, and security service management (in particular automated adaptation). The tool functions rely on a comprehensive application model.

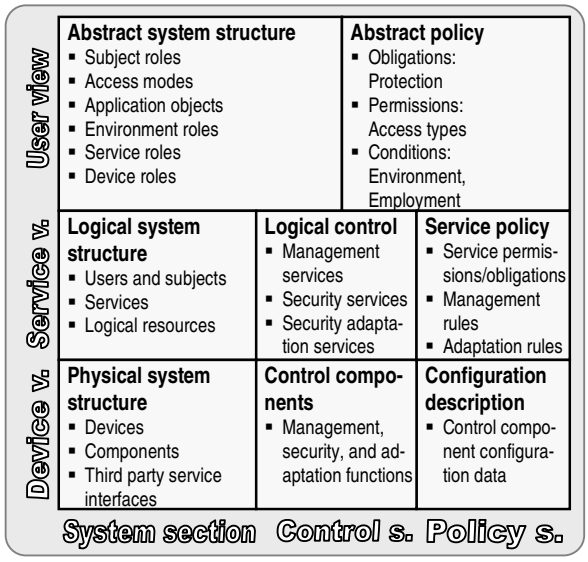

Fig. 4: Model structure

The model has the form of a graph which defines an object instance diagram. The nodes are object instances representing the relevant physical and logical building blocks of the application. The edges represent associations between object instances. The nodes are arranged on a plane which is structured into three rows and three columns (cf. Fig. 4).

The model *rows* correspond to three levels of abstraction. The *user view* models the logical structure of an application as it is relevant to its users. It focuses on abstract application objects and operations. It particularly includes the *GRBAC* model. The *service view* models the logical implementation structure focusing on services and service usage relations. It comprises the three service layers of an application's architecture (cf. Fig. 1). The *device view* corresponds to the physical implementation structure focusing on devices, software components, and service interfaces. It corresponds to the computing and communication layer of the application architecture.

The model *columns* correspond to three sections. The *system section* on the left contains the elements which deal with the proper application objects and their implementation. The *policy section* on the right represents the policies and rules which guide the automated management functions. The *control section* in the middle covers the two lower layers only. It contains those services respectively software components and interfaces which perform the security functions and automated management functions.

Due to the comprising model information, the tool can check the consistency of application configurations, can propose suitable modifications, can perform the security analysis, and in particular can automatically derive those security service and management configurations, which enforce the given abstract high level policies defined in terms of *GRBAC* and represented in the model's user view policy section. From that information the tool derives firstly the refining logical control system and its service policy (i.e., the model's *service view control section* and *policy section* are generated by the tool). Secondly the tool derives the physical control system and its configuration parameters

(i.e., the model's *device view control section* and *policy section* are generated by the tool). Finally, backend functions evaluate the *device view policy section* and generate configuration description files.

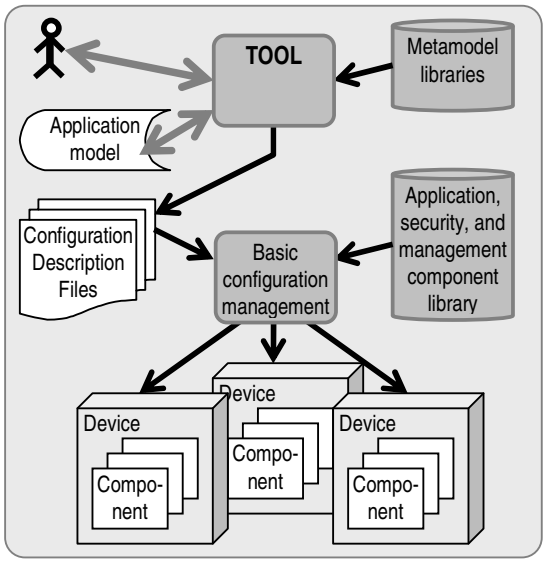

Fig. 5: Automated security management

The tool is an integral part of our overall security management approach which is depicted in Fig. 5. Besides of the *tool*, the *application model*, and the *interactive modeling* (upper left section), the figure refers to the *metamodel libraries* (upper right section). The libraries provide predefined object classes, from which the model objects are instantiated. Moreover they provide graph rules which control the analysis of the system section and the derivation of the policy refinements. The box in the center of the figure depicts the general *basic configuration management* of the *SIRENA* framework which supports the deployment of the application and thus performs the transition from design time to runtime. The deployment of the security and security management components is controlled by the *configuration description files* produced by the tool. The implementing software components are provided by the *component library*. The lower section of the figure depicts a runtime scenario where *devices* support and execute *software components*.

The described model-based derivation of policy-enforcing system configurations is utilized for the *adaptation* of security service employments to changing environments as follows. First starting point is the *abstract high level policy* in the *user view* of the model which contains the *relevant security requirement profiles*. Second starting point is the model's *device view system section* which represents all *possible implementation scenarios*. Based on that information, the tool derives a set of security system configurations, namely for each implementation scenario one corresponding configuration. All configurations are represented in the model's *device view control section* and *policy section*. Based on that information the generation of the configuration description files can consider all possible runtime scenarios and their transitions. At runtime, configuration management components detect environment change events and perform the corresponding prepared configuration transitions. In this way the adaptation is automatically planned at design time and efficiently performed at runtime.

VII. Concluding remarks

The described security management approach provides tool-assisted security analysis and security service planning for embedded networked applications. Moreover, it extends the applications with automated management components which provide adaptation to changing environment and configuration conditions at runtime. We outlined here the approach as a whole and focused on the roles of security analysis, high level policy definition, and application model. Details of the special model structure and descriptions of the diverse adaptation functions will be published elsewhere. Currently we integrate the tool and the functional components into the specific context of the *SIRENA* project where we closely cooperate with the developers of the general application management system. In fact, security management and general application management will use the same tool, and the security adaptation functions will heavily rely on the general management's configuration-monitoring and reconfiguration functions.

The work described herein was funded by the German Federal Ministry of Education and Research (BMBF) within the ITEA-SIRENA project (01ISC09G).

VIII. References

[1] R. Baskerville, "Designing Information Systems Security", Wiley & Sons, Chichester, 1988.

[2] Michael J. Covington, Matthew J. Moyer, and Mustaque Ahamad, "Generalized Role-Based Access Control for Securing Future Applications", in: 23rd National Information Systems Security Conference, Baltimore, October 2000.

[3] D. Ferraiolo et al., "Proposed NIST Standard for Role-Based Access Control", ACM Transactions on Information Systems Security, Vol. 4, No. 3, August 2001.

[4] D. Ferraiolo and R. Kuhn, "Role Based Access Control", in: Proceedings of the 15th National Computer Science Conference, 1992.

[5] P. Herrmann and H. Krumm, "Object-Oriented Security Analysis and Modeling", in Proceedings of the 9th Int. Conference on Telecommunication Systems, pages 21-32, ATSMA, IFIP, 2001.

[6] ISO/IEC, "Common Criteria for Information Technology Security Evaluation", International Standard ISO/IEC 15408, 1999.

[7] I. Lück, C. Schäfer, and H. Krumm, "Model-based Tool-Assistance for Packet-Filter Design", in: M. Sloman, E. Lupu, J. Lobo (Eds.), Policy 2001, LNCS 1995, pp. 120-136, Springer-Verlag, 2001.

[8] I. Lück, S. Vögel, and H. Krumm, "Model-based configuration of VPNs", in R. Stadtler, M. Ulema (eds.): Proc. 8th IEEE/IFIP Symposium NOMS 2002, IEEE, pages 589-602, 2002.

[9] J. Moffet and M. Sloman, "Policy Hierarchies for Distributed Systems Management", IEEE Journal on Selected Areas in Communications, 11, 9, 1993.

[10] M. J. Moyer and M. Ahamad, "Generalized Role-Based Access Control", in: IEEE Proceedings of the 21st International Conference on Distributed Systems, Mesa, April 2001.

[11] *SIRENA: Service Infrastructure for Real time Embedded Networked Applications*, Project in the European Framework ITEA, see URL: http://www.*SIRENA*-itea.org (2003).

[12] M. Sloman, "Policy Driven Management for Distributed Systems", Journal of Network and Systems Management, Vol. 2, No. 4, 1994.

[13] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest, "Proxy-based security protocols in networked mobile devices", in: Proceedings SAC, ACM Press, pages 265-272, 2002.