Integrating Security Policy Design in the Software Design

Cristian Oancea

10.11.2003

Abstract

Security is an integral part of most distributed modern software systems, but is still not considered as an explicit part in the development process. Security mechanisms and policies are generally added to existing systems as an afterthought, with all the problems of unsatisfied security requirements, integration difficulties and mismatches between running system and the design models. We propose to integrate the design of application-oriented access control policies early into the system's development process. The standard language for modeling the design of systems the Unified Modeling Language (UML), is used to specify access control policies. Within the integration we will develop extensions of the UML model support the automatic generation and verification of a access control policy to configure a distributed component-based for view-based access control.

1 Introduction

Security aspects are inherent in the distributed modern software systems that are used in untrusted environments. Yet there is little systematic support for software engineers who need to produce secure software, this is especially true for the distributed software design process.

There are at least two reasons for the lack of support for security engineering. The first reason is that security requirements are generally difficult to analyze and model. A second important reason is the lack of developer acceptance and expertise for secure software development. Acceptance is a problem because for software developers, security interferes with features and with time to market. Expertise is a problem because security policies are generally specified in terms of highly specialised security models that are not integrated with general software engineering models. This lack of integration is also detrimental to developer acceptance.

Security issues that must be addressed during design and development phases are not restricted to avoiding software vulnerabilities, such as buffer overflow errors, and meeting security requirements that were explicitly expressed at the beginning of the development process. In order to ensure that software can be properly installed, configured, operated, and managed by administrators, designers and developers must produce appropriate documentation of the application policies that they have designed and implemented. This documentation

must be sufficiently abstract so that security administrators can manage the policies without knowledge of application-internal design details.

The security requirements that are predominant in object distributed environments (just as in most other commercial, non-military applications) are integrity requirements that can be addressed by employing access control.

Because between the client and the shared resource communicate through an insecure medium of communication, like Internet, the need to enforce an access control policy is vital for the organisation that manages the shared resource, in order to protect its business. In order to be enforced, the requirements need to be specified in a access control policy.

The Unified Modeling Language (UML) is "one consistent language for specifying, visualising, constructing, and documenting the artifacts of software systems". In recent years, UML has become a standard for object-oriented modeling in the field of software engineering.

Specification of the security policies using UML will ease the understanding of security requirements, so the developer acceptance for security aspects will grow. In this way, security problems will be earlier detected in the system development process and the costs of removal will be smaller as the security aspects would be separately added.

Recent research concerns the integration of security engineering into the software development process [Jür02, LBD02]. These researches are focused on the UML modelling of the standard Role Based Control(RBAC) [SCFY96] models.

View-based access control (VBAC) is an access control model specifically designed to support the design and management of access control policies in object-oriented systems [Bro01a, Bro02] that provides an high-level language construct, the *view*, a typed grouping concept for the access rights, that helps writing the security specifications and allows static-typed checking to ensure the consistency of the specifications [Bro00].

We explore the possibility of representation of VBAC models using UML. Our main objective is to provide an object-oriented approach to direct future work in view-based system analysis and design of the access control policies for distributed systems and to validate the security policies against the system's UML model. In order to model, validate and generate the security policies we will extend the UML notation with specific constructs that are defined into UML profiles, which is the standard extension mechanism of the UML.

2 View Based Access Control and RACCOON Infrastructure

The *View*, one of principal features of VBAC, allows the description of fine-grained access rights, which are permissions or denials for operations of distributed objects. Further views defined on objects are assigned to principals, i.e., to individual subjects or user roles, and a principal is allowed to invoke an operation of an object if it has a view on the object with a permission to call the operation, otherwise the access is denied if either a denial is specified or no permission is found in the assigned views hierarchy.

Views are defined in the *View Policy Language* (VPL) as part of a policy

design document, which is a product of the design stage in the development process. Furthermore, VPL defines view extension, so that an extending view inherits all access rights of the base view. Views can statically be restricted such that they can only be assigned to specific roles and views can be declared to be *virtual*. Virtual views have empty bodies. To specify automatic changes in the security state, VPL defines *schemas*. A schema defines triggers for the automatic assignment and removal of views to principals. These automatic changes in the protection state are especially important in dynamic environments with frequent rights changes that are required by application policies. Performing such changes manually would be too cumbersome to be practical.

View-based access policies are type-checked with a language compiler that catches a number of specification errors, such as access rights that cannot apply to a target object type and conflicts between different views that both allow and deny an operation on an object.

The deployment and management infrastructure designed for this approach is called *Raccoon* [Bro01a, Bro01b]. A deployment tool processes policy descriptors and stores static view and role definitions in repositories that can be managed using graphical management tools. At runtime, role membership is represented by digital certificates issued by a role server. Access decisions are made locally in the server processes that host application objects. These decisions are made on the basis of policy information that is supplied by policy servers, which rely on the deployed policy information.

3 Access Control Policy Design

The software systems have a long life cycle and the specifications and models should be technologically neutral. Platforms are subject to change over time. Most of all, they change at different, typically higher, rates than the "higher-level" models of the system that tend to grow increasingly independent of the target platforms. Enriching system models with enough design details makes it possible to switch the technology without an complete system redesign effort.

This idea is applied into the Model Driven Architecture(MDA), that uses high-level models to describe the systems. The models are described using UML models at various levels of abstraction each emphasizing certain aspects or viewpoints of the system. The system can be be deployed to one or more platforms, either used alternatively, or in conjunction.

The primary artifacts of the UML can be addressed from two perspectives: the UML definition itself and how it is used to produce project artifacts. The UML definition consists of the semantics and extensions. The UML semantics define the language syntax, semantics, rules, and constraints. User-defined extension is a predefined set of stereotypes, tagged-values, constraints, and notation icons that collectively extend and tailor the UML for a specific domain or process. The development project artifacts concern about the modeling technologies including use case modeling, object (static) modeling, and dynamic modeling.

The standard meta language defined by the OMG is Meta-Object Facilities (MOF) [OMG] the in which other modeling languages can be specified. The advantage of the MOF-approach is to make the definition of (meta)-models independent of the concrete application domain of the models and to provide a

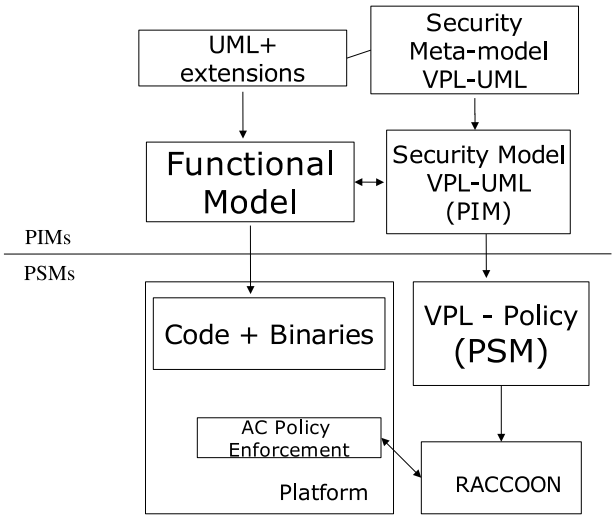

Figure 1: Security Model

concise and unique set of concepts for the definition of metamodels. Moreover, multiple meta-models can be managed by the MOF and relations between meta-models can be utilized as a basis for a transformation of models. We combine the MOF models with UML profiles concerning a specific platform or a specific application domain to get a usable and readable notation of MOF models.

According to MOF, models are instances of metamodels. A metamodel may make it possible to describe properties of a particular platform. In this case, the models that are instances of such a metamodel are said to be platform-specific with regard to this platform. Models that describe a system at a level of abstraction that is sufficient to use all their contents for implementing the system on different platforms are referred to as platform-independent with regard to these target platforms.

Models and, by implication, their metamodels, may have a semantical relation to one another, for example if they describe the same system for the same platform at different levels of abstraction. It is of course desirable to have mappings between those different but related models performed automatically. This makes it possible to express each aspect of a system at an appropriate level of abstraction while keeping the various models in sync. A mapping between models is assumed to take one or more models as its input and produce one output model. Again, each of these models is an instance of a metamodel. The rules for the transformation that is performed by the mapping are described in a mapping technique. They are described "at the metamodel level" in such a way that they are applicable to all sets of source models being instances of the source metamodels.

A platform is the specification of an execution environment for models. A platform is only considered in the context of MDA if there is at least one realization of it. A realization can be seen as the implementation of the specification that the platform represents.

An access control model is specified by a platform and application domain

independent MOF meta-model. On the one hand, this metamodel can be instantiated in an application domain, i.e., it specifies then a model for an access control policy specific for an application. On the other hand, the meta-model can be refined to a platform specific MOF meta-model (e.g., CORBA, J2EE, SOAP). Combining both, application domain instantiation and platfrom specific refinement, results in a specific application model having a customized application access control policy that is implemented in a specific platform.

In the following we give an overview of our model-driven approach to develop a VPL policy descriptor for a specific application running on a specific platform. Figure 2 depicts the general idea. As first step for designing the access controll policies for an application, the *system designer* creates a platform independent model, a *VBAC-PIM*. This model specifies the policies for accessing the interfaces that contain the operations of the system.

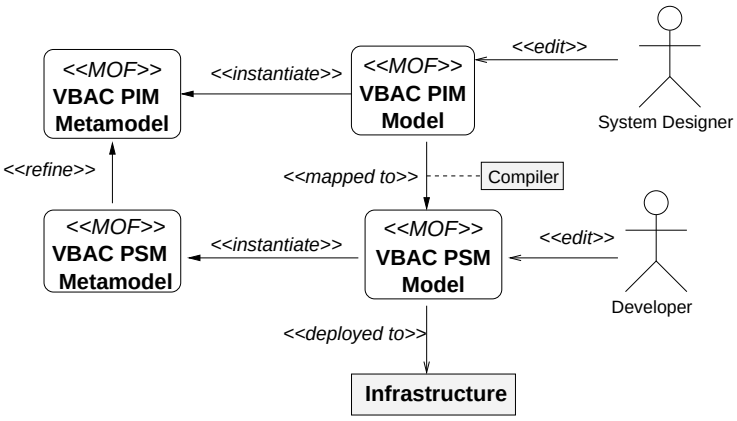

Figure 2: *MDA VPL models.*

After the system specification phase, the development group chooses the technologies for their implementation. The platform independent models are mapped to platform specific models for the chosen platforms. To map the models related to access control, the VBAC-PIM is compiled to a *VBAC-PSM*. We will show later how this is done for the CORBA and the EJB platforms.

This automatically generated VBAC-PSM is not complete. Technological parameters, such as name servers or proxy settings, cannot be read out of the VBAC-PIM. A *developer* has to add these missing parts. Finally the VBAC-PSM is deployed and enforced by the specific infrastructure.

The PIM–Metamodel (PIM-MM) specifies the elements provided by VBAC. Since VBAC is developed for distributed object systems, the PIM-MM is intended for a platform based on objects (e.g., CORBA).

The PIM–MM specifies mainly roles, views and schemas. The role attributes *minCard* and *maxCard* define the minimal and maximal number of principals that must resp. can play the role. Roles can be ordered in a hierarchy (extends-association), may exclude each other, i.e., one principal cannot play two (or more) roles in mutual exclusion relation at the same time, and may require other roles as a prerequisite. Roles may be assigned to several principals and principals may play several roles at the same time.

The UML model is used to generate a CORBA security specification in VPL that is deployed together with the CORBA application. The integration of the security policy design into the software development process with UML uses existing UML extension possibilities. The diagrams (use cases extended by notes, view diagrams, role diagram, and schema diagrams) can be drawn using existing UML tools.

4 Future Work and Open Questions

Using early integration of the security requirements in the UML design models,it is possible to identify in the models unwanted resp. forbidden accesses which can be corrected before the implementation phase of the project. The detection of security conflicts in the UML models should be supported by appropriate tools.

For the development of the design concepts in the software development process the Common Criteria will be introduced and in this way we can make the evaluation of security requirements.

Identification the security requirements and Integration in the UML designing proccess. In order to be able to enforce the access control policies in an large enterprise application, we have to integrate the RACCOON Repositories for working with other distributed platforms(EJB,SOAP Web Services, .NET).

In our previous case studies we have identified the need of integration of VPL views generation resp. construction with sequence diagrams design, in this way the access control policy will be closer integrated into the application model.

We have to take into account the security policy modeling for the large applications that incorporate more then one distributed platform, like Web Services and CORBA resp. EJB. The access control enforcing infrastructure should be able to handle the technologies details regarding name resolution, interface description and others.

References

[Bro00] Gerald Brose. A typed access control model for CORBA. In *ESORICS*, pages 88–105, 2000.

[Bro01a] Gerald Brose. *Access Control Management in Distributed Object Systems.* PhD thesis, Freie Universität Berlin, 2001.

[Bro01b] Gerald Brose. Raccoon — An infrastructure for managing access control in CORBA. In *Proc. Int. Conference on Distributed Applications and Interoperable Systems (DAIS).* Kluwer, 2001.

[Bro02] Gerald Brose. Manageable Access Control for CORBA. *Journal of Computer Security*, 4:301–337, 2002.

[Jür02] J. Jürjens. UMLsec: Extending UML for Secure Systems Development. In *Proc. of UML 2002*, number 2460 in LNCS, pages 412–425. Springer, 2002.

[LBD02] T. Lodderstedt, D. Basin, and J. Doser. SecureUML:A UML-Based Modeling Language for Model-Driven Security. In *Proc. of 5th Int. Conf. on the Unified Modeling Language*, number 2460 in LNCS. Springer, 2002.

[OMG] OMG. Meta-object facilities. http://www.omg.org/cgi-bin/doc?formal/2002-04-03. Specifications v1.4.

[OMG99] OMG. *CORBA 3.0 New Components Chapters, TC Document ptc/99-10-04*. OMG, October 1999.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[Sun00] Sun Microsystems. *Enterprise JavaBeans Specification, Version 2.0, Final Draft*, October 2000.