Model Driven Security Management: Making Security Management Manageable in Complex Distributed Systems

Dr: Ulrich Lang, Rudolf Schreiner

ObjectSecurity Ltd., St. John's Innovation Centre, Cowley Road,
Cambridge CB4 0WS, United Kingdom
{Ulrich.Lang | Rudolf.Schreiner}@objectsecurity.com

Abstract. Today, the challenge in security of complex distributed systems does not anymore lie in encryption or access control of a single middleware platform, but in the protection of the system as a whole. This includes the definition of correct security policies at various abstraction layers, and also in the unified and correct management and enforcement of the correct security policy at all relevant places in the system. The authors have learned in the development even of comparatively simple distributed systems that this is not possible anymore by a manual definition of encryption properties and access control rules. Human security administrators are not able to define all these fine grained rules with sufficient assurance, to distribute them to all Policy Enforcement Points and to check many log files or admin consoles. This is especially impossible in highly distributed and agile service oriented or data driven systems. In this paper we will illustrate the approach and architecture behind Model Driven Security Management and provide a healthcare regulatory compliance case study using our OpenPMF 2.0 technology.

Keywords: model driven security, security policy, security policy management, authorization management, identity & access management, XACML, security enforcement, service oriented architecture (SOA), regulatory compliance.

1. Introduction

The protection of distributed systems in large organizations is often a great challenge. In most organizations there are many applications based on different middleware platforms and there is also a large number of users. Today, the common approach to protect such systems is based on a security policy document. This document balances the functional business requirement of the application, the security requirements, legal and regulatory requirements and other factors like costs. This security document is written in human language and given to the IT staff. The IT staff now has to enforce the security policy using so called controls, for example encryption or access control systems like Virtual Private Networks (VPN) or firewalls. Typical enterprise security architecture today consist of an identity management system, e.g. used for Single Sign on, firewalls and VPNs for connection to remote sites or business partners, and application layer access control directly implemented in the application logic or, in the best case, using more or less sophisticated authorization and entitlement management systems.

This approach raises several issues, for example policy correctness and consistency, even for quite simple client/server or 3-tier applications How can be ensured that the high level policy is really correctly mapped to a high number of access control rules and configurations of security mechanisms at all times? Even if parts of the systems are administered in different departments or if administrators are absent? How can many different controls be managed in a uniform way? How can policy violations be detected and handled? Therefore, this approach was never really good; it was error prone, had high maintenance costs and required a lot of resources. But, nevertheless, many organizations considered it sufficient.

In the future, this approach will not be sufficient anymore. First of all, there is a paradigm shift in the development of distributed applications. Secondly, in many domains closed systems have to be opened to communication with the outside world. In particular, the complexity arises because of the need for business-driven IT agility, where IT policies and enforcement can change frequently, e.g. Business Process Modelling (BPM) driven Service Oriented Architecture (SOA).

And finally, regulatory and legal requirements demand a higher level of security, including the proof that the system is really sufficiently protected. Compliance with regulatory and governance standards is rapidly becoming one of the more pervasive hot topics of information security today because organisations have to expect large financial and reputational losses if compliance cannot be ensured and demonstrated. One major difficulty of implementing such regulations is caused by the fact that they are captured at a high level of abstraction that is business-centric and not IT centric. This means that the abstract intent needs to be translated in a trustworthy, traceable way into compliance and security policies that the IT security infrastructure can enforce. Carrying out this mapping process manually as described above is time consuming, maintenance-intensive, costly, and error-prone. Compliance monitoring is also critical in order to be able to demonstrate compliance at any given point in time.

Model Driven Security (MDS) [1] management is an innovative technology approach that can solve these problems as an extension of identity and access management (IAM) and authorization management (also called entitlement management). In this paper we will illustrate the theory behind Model Driven Security for compliance, provide an improved and extended architecture, as well as a case study using our OpenPMF 2.0 technology [2]. Model Driven Security management also helps achieve and demonstrate regulatory compliance.

Section 2 outlines today’s security management complexities; section 3 introduces the model driven security concept; section 4 illustrates OpenPMF 2.0, an implementation of the concept; section 5 presents a concrete example, and section 6 draws some conclusions.

2. Today’s typical security management complexities

One major difficulty of implementing security policies is caused by the fact that policies (and regulations) are expressed at a high level of abstraction that is organisation-centric, business-centric, information-centric, legal aspects centric and human-centric. Policies/regulations are often not IT centric, nor expressed in IT terms. This means that the abstract and high level policy requirements defined by the regulations need to be translated into compliance and security policies that the IT security infrastructure can enforce. Carrying out this mapping process manually is time consuming, maintenance-intensive, costly, and error-prone. An automated, reliable technology approach is required to solve these issues.

Another major difficulty is related to compliance monitoring. How can an organisation demonstrate sufficiently that it complies with these organisation-centric, information-centric and human-centric and legal regulations, security policies, and governance standards? And how can attempted compliance violations be detected early on and prevented before damage is caused? Doing this manually, as suggested by many survey-based compliance tools is too slow, costly, and error-prone. How can this be done in an automated way that is cost effective, timely, reliable, and automatic? Again, a suitable technology approach is required.

These two difficulties are particularly hard to deal with in distributed IT environments that are “agile”, i.e. get reconfigured regularly. The current architectural style for agile distributed software applications is Service Oriented Architecture (SOA). One concept behind SOA is to specify and manage application interactions in an abstract model that bridges the gap between the business (business processes, workflows, BPM etc.) and IT (platforms like web services, databases etc.). In such model driven SOA environments, enterprise architects specify workflows in Business Process Modelling (BPM) suites, which are used to orchestrate underlying modular IT services. One of the main benefits is that the IT environment can be reconfigured easily to reflect changes in the business. For security policy enforcement and monitoring, SOA agility poses a major complexity because security policies (both for compliance enforcement and monitoring) will have to be updated each time the IT environment gets reconfigured. Carrying out such policy updates manually each time the SOA gets reconfigured is clearly unworkable because this would be too costly, too slow, and too error-prone. So again, a suitable technology approach is required to automatically update compliance enforcement and monitoring whenever the SOA changes.

Another interesting related aspect is how trustworthy the link between the regulation or governance standard on the high layer of abstraction, and its IT enforcement and monitoring on the low layer of abstraction is. This trustworthiness is relevant for defence accreditation (e.g. Common Criteria [3]) and many other safety and assurance standards. This is an especially important factor for agile systems, where an accreditation approach of a static system (the normal way to achieve accreditation) is not possible.

3. Model Driven Security

Model Driven Security (MDS) [1] is an innovative technology approach that can solve these problems. MDS can be defined as the tool supported process of modelling security requirements at a high level of abstraction, and using other information sources available about the functional aspects of the system (produced by other stakeholders). These inputs, which are expressed in Domain Specific Languages (DSL), are then transformed into enforceable security rules with as little human intervention as possible. MDS explicitly also includes the run-time security management (e.g. entitlements/authorisations), i.e. run-time enforcement of the policy on the protected IT systems, dynamic policy updates and the integrated monitoring of policy violations.

In the first step of MDS, regulations and governance standards are modelled as a high-level security policy in a Model Driven Security tool (such as ObjectSecurity's OpenPMF 2.0). These models are then translated into low-level security policies that are enforced across the entire SOA environment (e.g. through local plug-ins integrated into the middleware or at a domain boundary). The local plug-ins also deal with the monitoring of compliance/security-relevant events. If tied into the SOA BPM suite and the SOA middleware (e.g. web application server), Model Driven Security can automatically update the compliance enforcement and monitoring whenever the SOA application changes.

Model Driven Security management (in the form implemented in OpenPMF 2.0 by ObjectSecurity) has the following benefits:

- Model driven security regulates information flows and resource access between different systems (or software services/components) and users in a fine-grained, policy-driven way across a potentially large, heterogeneous IT environment. Furthermore, it ensures that security policy updates caused by IT agility, i.e. changes to the IT environment, can be managed without a maintenance cost explosion. All this helps reduce the security management effort.
- Another core benefit is that Model Driven Security management helps align business security requirements (including regulatory and best-practice security requirements) and policy-driven technical IT security enforcement. This means that security requirements, which can be captured in an undistorted, abstract way close to human/business thinking, can be automatically transformed into technology-centric IT security enforcement. This reduces the cost and effort of security policy definition and maintenance. The automated technology approach also improves the traceability from requirements to enforcement and improves assurance, because human administration errors are minimised.
- Cost/effort savings can be significant. In various projects, the authors showed that thousands of rules can be generated and maintained automatically, which saved several person weeks of effort (on the flipside, this saving is lessened a bit by the effort of developing the model transformations and high-level requirements model).

- In addition, the Model Driven Security approach can tie security requirements into the overall business enterprise architecture, which ensures that the needs of the business are reflected.

MDS is an emerging hot topic. There is little doubt that some form of model-driven approaches in general will become mainstream over the coming years (in whatever form). This is because IT complexity in today's complex, interconnected, ever-changing, ever-faster world has become unmanageable, and it has always been human nature to try to find an abstraction in order to be able to easier deal with that complexity.

On the other hand, pervasive changes to IT and business staff behaviour and technology has always posed a formidable adoption hurdle. However, industry analysts such as Gartner [4] forecast model-driven approaches to become mainstream within five years, based on high-profile, big-vendor pushes (e.g. Microsoft "Oslo" [5]). Model Driven Security will then be a feature with obvious benefits in the overall architecture.

Irrespective of that, the adoption of Model Driven Security does not have to wait for model-driven approaches to become mainstream. Instead, a non-intrusive gradual adoption roadmap is possible, starting with authorisation management as a policy-driven add-on to today's identity management (IdM) deployments. While not model-driven, such authorization management (integrated with IdM) can help bridge the time until full Model Driven Security gets adopted.

The authors see the most complete deployment of Model Driven Security as a scenario where a Model Driven Security tool is used as an enterprise-wide security management tool that ties in with enterprise architecture (incl. BPM) , model-driven integration, model-driven engineering. Such a pervasive deployment would do away with the many different and incompatible ways of managing security infrastructure, and thus provide significant benefits, including cost-saving and manageability. An additional benefit of such a deployment is that the end customer "owns" the security policy model, and is therefore in a position to outsource most of the IT and IT security infrastructure "plumbing" to vendors and integrators without losing control and in-house security expertise.

4. OpenPMF 2.0 Model Driven Security Management

OpenPMF [2] is a framework for the unified definition, management and enforcement of security policies in complex distributed systems. It allows a definition of security policies in a human readable Policy Description Language (PDL) [6], the central management and automatic distribution of security policies to Policy Decision Points (PDP) Policy Enforcement Points (PEP) collocated with the middleware and application. It also includes a central management of security events, for example policy violations. OpenPMF can be used to protect different communication patterns. Enforceable security policies are described in PDL and automatically distributed to the PDP/PEPs in the different middleware platforms (including Web services, JMS, CORBA, CCM, DDS, MOM etc) or the applications. During runtime, the PEPs control the invocations or the data flow. If a policy violation is detected, a notification is sent to the central admin console. Practical evaluations and tests, and using OpenPMF in real world applications, showed that it really improved policy management and enforcement a lot. There was one, central consistent security policy for the whole application, and a single administration console. It was not necessary anymore to manually configure different security mechanisms and to check multiple log files. But the policy definition still turned out to be difficult.

Initially, when the authors developed OpenPMF, they expected to write the policies in the PDL language. For simple and static client/server applications this works indeed well. ObjectSecurity's OpenPMF 2.0 security management technology is a significant extension and enhancement of the OASIS eXtensible Access Control Markup Language (XACML) [7], which is a standard architecture and XML based syntax for transferring authorisation rules called within the web services world. XACML includes Policy Administration Points (PAPs), Policy Decision Points (PDPs), Policy Information Points (PIPs), and Policy Enforcement Points (PEPs). OpenPMF 2.0's extended and improved architecture also includes model-driven policy management points and runtime central monitoring points, two critical features to make security policy management manageable. While XACML has been originally advocated as the solution for fine-grained authorisation management (also called entitlement management), it is evident now that XACML (and authorisation management) solves more the security policy interchange challenge than the security policy definition and management challenge. This is because the XACML architecture assumes that the (typically many) fine-grained technical enforcement rules are defined and managed at the low level of abstraction of IT enforcement. Specifying and maintaining thousands (or more) rules manually in an agile, ever-changing IT environment is clearly impossible. Model Driven Security can be seen as the missing piece at the top of the architecture that makes security authorisation management manageable by generating these low-level rules from abstract requirements.

In contrast to XACML, PDL is human writable, with a syntax similar to the CORBA Interface Description Language. But in more complex systems (e.g. the AD4 ATC simulation system [8]), it turned out that the manual policy definition is infeasible with a sufficient level of assurance. The exact problem, as turned out, was not to write down the single rules of the policy, but understanding the system and all its interactions, both business logic and infrastructure, in sufficient detail. OpenPMF was good at the management and correct enforcement of security policies, but the definition of correct policies was still an issue.

The key to policy correctness lies in the full integration of security into software development. Today, it is very common to use Model Driven Engineering (MDE) for the development of complex distributed systems. Systems, their components/services and interactions, are modeled using a generic modeling language like the Unified Modeling Language (UML), Business Process Modeling Notation (BPMN) or a Domain Specific Language (DSL) tailored to specific application domains. To allow the common interpretation of models, these modeling languages are expressed in a so called "meta meta model", for example the OMF Meta Object Facilities or Eclipse

Ecore[1]. The models, as normally used today, describe the functional aspects of the system to develop, for example data types, service and interface definitions, service interactions and workflows in business processes, deployments of components and services onto hosts and networks, and also users. From these models, so called model transformations automatically generate many programming artifacts and configuration files. For example, from a service or interface model it is possible to generate a CORBA interface description in CORBA IDL, a Web Services service description in WSDL, or the Java classes of a Web Services implementation.

For the definition of the security rules and configurations, this functional model is most important. It provides all the detailed information which is required for the rule definition and hard to fully understand process for human administrators. But this is still not sufficient to define the rules, since a high level security policy is missing.

In the first version of Model Driven Security as used in the AD4 ATC simulation system [8], this high level security policy was hard coded in the transformations to generate the rules and configurations. This was good enough for the concrete task, but in general flexible security policies are required, depending on the organization's enterprise wide security policy. To define a high level security policy, a specific DSL for this purpose, the Security Policy Language (SPL) is used, normally based on the same meta meta model as the functional model. In SPL, the high level security policy is described. This includes for example the definition of the Quality of Protection ("All communication has to be protected at level X", or "All protection outside the own network has to be protected at level Y"), access control ("Accountants are allowed to used the accounting system during working hours") or information flow policies ("For all information flow, Mandatory Access Control (MAC) has to be applied")[2].

Now, all information required for the generation of the security rules is available: We have the detailed functional model of the system and a high level security policy, expressed in SPL. From both models, model transformations are now able to generate all artifacts relevant for security, for instance the access control rules expressed in PDL or the command line options for encryption (Fig.1). For example, in information flow based on a publish/subscribe middleware, the rules for the publish and subscribe operations are generated, and also for the MAC properties of the data distribution.

[1] This description of Model Driven Engineering is simplified, but it is sufficient to understand the concepts of Model Driven Security.

[2] Currently, we have a meta model for SPL, but no textual representation. Therefore we use free text for the description of the policies.

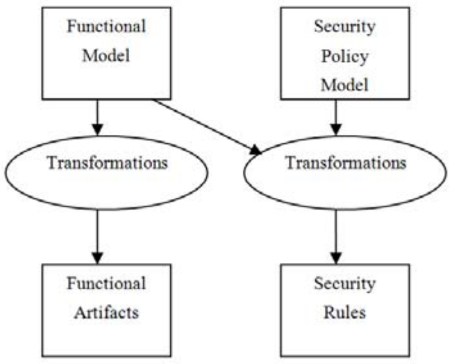

Figure 1: Generation of High Assurance Security Rules from Models

Instead of hundreds or thousands of very fine grained rules, a policy in SPL consists of a very low number of rules with a high abstraction level. For example, an appropriate security policy could define the encryption strength to use for authentication and message protection, and the rule that all communication defined in the functional model has to be allowed both for invocations and events. For humans, such short and high abstraction layer policies are much more accessible.

Model Driven Security solves many of the problems of manual policy definition, because the security rules and configurations are always in line with the functional behavior of the system. First of all, MDS greatly improves the assurance of the policy, because the policy is generated from well tested or verified transformations. This trusted transformation is also superior to policy verification, because it takes the functional behavior of the system into consideration, instead of just checking a policy for internal inconsistencies. Model Driven Security also makes security in SOA manageable, because whenever the agile applications based on interacting services are modified, the security policy is adapted as well. This not only ensures a correct policy but also greatly reduces the maintenance effort.

5. Model Driven Security Healthcare Example

This section illustrates how one exemplary security-related high-level healthcare regulatory requirement (HIPAA) is translated into enforceable authorisation rules (another example can be found in [6]). The exemplary (particularly intuitive) healthcare security/privacy requirement is simple: "*Every doctor is only allowed to access the patient record of the patient they are currently treating (unless the patient is treated in a crisis context, or the patient consents).*" This requirement is stated in an abstract way, independent of the particular IT environment, patient identity, doctor identity, doctor-patient relationship, IT environment etc. In line with the OMG Model Driven Architecture (MDA) framework, we call these requirements "undistorted" by the particular deployment. This requirement can be captured in a modelling tool in a customised, customer-specific way.

A crucial dimension of Model Driven Security is that other system models are reused in order to infer the authorisation rules for the particular deployment. There are potentially many other sources for system models, such as Business Process Management System (BPMS) tools which are for example used by enterprise architects to capture the processes of a business and map them to the IT environment (e.g. SOA BPEL/web services). OpenPMF 2.0 has been designed to co-exist with modelling tools (both BPM and MDA) inside the same Eclipse IDE installation. Model Driven Security also needs to have access to the information about which particular participants exist, such as doctors and patients (the "who" in authorization rules). In many organisations, this information has been collected in a centralised identity management solution (IdM), which typically do not provide a means of specifying and managing (potentially many) fine-grained access rules (the "what" and "how" in authorization rules). Model Driven Security can be seen as an extension for such already-deployed IdM solutions, and OpenPMF 2.0 reuses such information sources.

This high level regulatory requirement can then be transformed into potentially very many deployment-specific IT-centric message protection and authorisation rules at different layers. With the model transformations and the models in place, many (potentially hundreds of thousands) of specific, fine-grained authorization rules can be generated from a single high level rule, with the push of a button. For example, generated rules can contain the following information: *allow information flow if "caller X.509 cert. id doctor1" via "firewall IP…" calling "file patient1" on "database IP …" and from "hospital IP …" and "doctor1 is treating patient1" and/or "patient1 crisis"*.

The next step is to distribute the generated authorization rules across the distributed, heterogeneous IT environment to enable runtime security enforcement of information flows through local Policy Enforcement Points (PEPs) on the systems that need to be protected.

To close the security management loop, it is crucial to monitor security-relevant activities across the distributed IT environment in the central policy. As an optional, additional, deeper level of defence, OpenPMF 2.0 pushes its policy-based alerts into an intrusion detection system for attack pattern mining.

6. Conclusion

In their practical work, the authors have learned that today the challenge in the protection of complex distributed systems does not lie in the encryption of communication or access control for specific middleware platforms anymore. The main issue is the correctness of all the configurations of the security controls like encryption, logging and access control of the different security mechanisms and platforms dispersed all over the system, their distribution and updates, their correct enforcement and also the processing of logging information, e.g. policy violations. As our previous projects have shown, human security administrators are for example not anymore able to define access control rules with sufficient assurance. The complexity is especially unmanageable in agile service oriented and data driven systems, were the security rules always have to be adapted to the modifications of the systems, e.g. reconnections of services.

In order to better protect such systems, and to lower the effort of protection over the whole system life cycle, we improved two aspects: The definition of correct security polices at all layers, and the correct management and enforcement of these security policies in the system as a whole. First of all, we embedded the definition of security policies into the model driven development processes, which are commonly used for the development of complex systems. This approach, called Model Driven Security, generates a large number of low abstraction layer rules and configurations from a very low number of high level security policy rules. Then we used our OpenPMF Policy Management Framework to manage these rules, e.g. to distribute them to Policy Enforcement Points in the systems, and also to correctly enforce them.

The benefits of Model Driven Security were demonstrated in the so far biggest application of Model Driven Security, an experimental secure ATC system. In this system, with over a dozen service types, about 50 service instances running on 10 hosts on three sites, MDS automatically generated a security policy of more than 1000 rules from the systems' functional model. Whenever the system was modified, e.g. by changing the services, their interactions or the deployment of services on hosts, the security policy was automatically adapted accordingly. This ensured that the system was always protected according the high level security intent. The policy maintenance effort during these modifications was reduced to almost zero.

In this paper we also illustrated the approach, architecture, benefits, and adoption roadmap for Model Driven Security in the context of (regulatory) compliance. We described why our implemented policy management architecture is a significant extension and improvement to current industry standards such as XACML. Details of our implementation are provided as part of a healthcare regulatory compliance case study. The case study is based on ObjectSecurity's OpenPMF 2.0 security management product, which is the leading Model Driven Security technology in the market. The paper also discusses the general industry direction with respect to model-driven approaches.

7. References

1. Model Driven Security web site, www. modeldrivensecurity.org
2. ObjectSecurity: OpenPMF 2.0 Model Driven Security Management, www.openpmf.com
3. Common Criteria Portal website, www.commoncriteriaportal.org
4. Gartner website, www.gartner.com
5. Microsoft, Oslo SOA & business process website, www.microsoft.com/soa/products /oslo.aspx
6. Lang, U., Schreiner, R.: Integrated IT Security: Air-Traffic Management Case Study. ISSE 2005 Conference Budapest, Springer Proceedings, 2005
7. OASIS Consortium: XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0, 1 Feb 2005
8. Schreiner, R., Lang, U., Ritter, T., Reznik, J., Building Secure and Interoperable ATC Systems, Eurocontrol INO Workshop 2006