Published at the CALIBRE Open Source Conference in Dijon, France, 5.-9.07.2005 – © 2005 ObjectSecurity Ltd.

OpenPMF:

Using Open Source for Security Policy Integration and Intrusion Detection in Heterogeneous Distributed IT Systems

Dr. Ulrich Lang, Rudolf Schreiner,
{ulrich.lang | rudolf.schreiner}@objectsecurity.com

Abstract

Most organisations today need to maintain and periodically migrate a heterogeneous distributed IT infrastructure to more modern platforms. This is typically a time-consuming, expensive, and error-prone process. The same problem also applies to IT security – security technologies and policy management consoles are changed periodically. We use software modelling concepts to ease the maintenance and migration efforts of distributed systems infrastructure and in particular distributed systems security (e.g. for CORBA, CORBA Components, EJB, Web Services, .NET). The use of Open Source software can simplify this task further because infrastructure technologies can be flexibly adapted and integrated with existing technologies. In this paper, we will discuss the challenges and benefits of using software modelling and open source software help make distributed applications and their complex security policies weather modifications in the underlying infrastructure technologies. We also present OpenPMF, our innovative Open Source security framework and implementation which enables centralised, technology-independent security policy management and intrusion detection for distributed heterogeneous systems. The paper argues that the concepts of model-driven software development can be successfully applied to security, and that this approach yields a number of benefits. Moreover, we claim that Open Source software can support this process because customization and integration problems are mitigated when the source code is available.

Introduction

The heterogeneous networked IT landscape that has evolved in many organisations over the decades makes application integration and migration a difficult task. The problem is aggravated by the fact that new and incompatible software platforms are developed every couple of years. In the 1990’s, the software industry believed that eventually one common standards-based software platform for distributed systems (e.g. CORBA or EJB) would emerge that enables application interoperability. A number of Open Source implementations of such middleware emerged, e.g. for CORBA (MICO and JacORB) and CORBA Components (Qedo and OpenCCM). Today, many of these Open Source middleware platforms boast production-grade quality.

A couple of years ago it was finally accepted that there will probably be a periodically emerging “next best thing” (that is incompatible to the old platforms) for the foreseeable future. Consequently there is a need to integrate applications across heterogeneous platforms and to migrate existing applications to new technologies. To do this manually is expensive, labour-intensive, time-consuming, and error-prone because of the significant differences in protocols, APIs etc. Model driven software development tackles this problem by first modelling the application logic in a technology-neutral way and by then mapping this model to the technology.

Security faces a similar problem because there are many different platforms and security technologies that need to be integrated and managed. Such an infrastructure makes it hard to enforce and administer a uniform, coherent, organisation-wide security policy. Typically, many insular security solutions are put in place and administered separately with differing semantics by administrators who are only concerned with their part of the overall IT landscape. The result is often a mishmash of conflicting, redundant, and incoherent policies. Moreover, it is often unclear if and how the abstract organisational security policy has been enforced adequately by the infrastructure. Intrusion detection (called policy violation detection in this paper) is also difficult in such systems because the information that needs to be analysed is on many different layers and is often not easily accessible.

This paper illustrates how model driven concepts can be applied to security, and presents OpenPMF, our technology-neutral, flexible security policy management framework. OpenPMF integrates the security policies enforced by heterogeneous technologies into a central technology-neutral policy repository. This way, consistent and optimised policies can be specified in a human-readable, unified fashion, and can be reused in the face of changing underlying technologies. OpenPMF also includes a prototypical policy violation detection daemon that collects and analyses information from the underlying IT infrastructure. As a result, OpenPMF minimises the total effort of security administration and increases security by reducing policy complexity, policy redundancy, policy inconsistency, and policy maintenance.

OpenPMF is available as Open Source software at www.openpmf.org. OpenPMF has been designed to integrate with almost any distributed systems platform and security technology. It has a very modular design with well-defined interfaces that allow flexibly replaceability and code reuse. It has been successfully integrated with CORBA, CORBA Components, Enterprise Java Beans, a public key infrastructure, an authorisation token server, an LDAP directory, an application layer firewall. Other technologies are added by the day (e.g. .NET Remoting and intrusion detection).

One of the advantages of using an Open Source software licensing model for such a technology is that users can be sure that they can integrate their (legacy and proprietary) infrastructure into the security framework in the future. Moreover, although we provide high-quality, reliable, and cost-effective support contracts for OpenPMF, there is no explicit vendor lock-in compared with proprietary software.

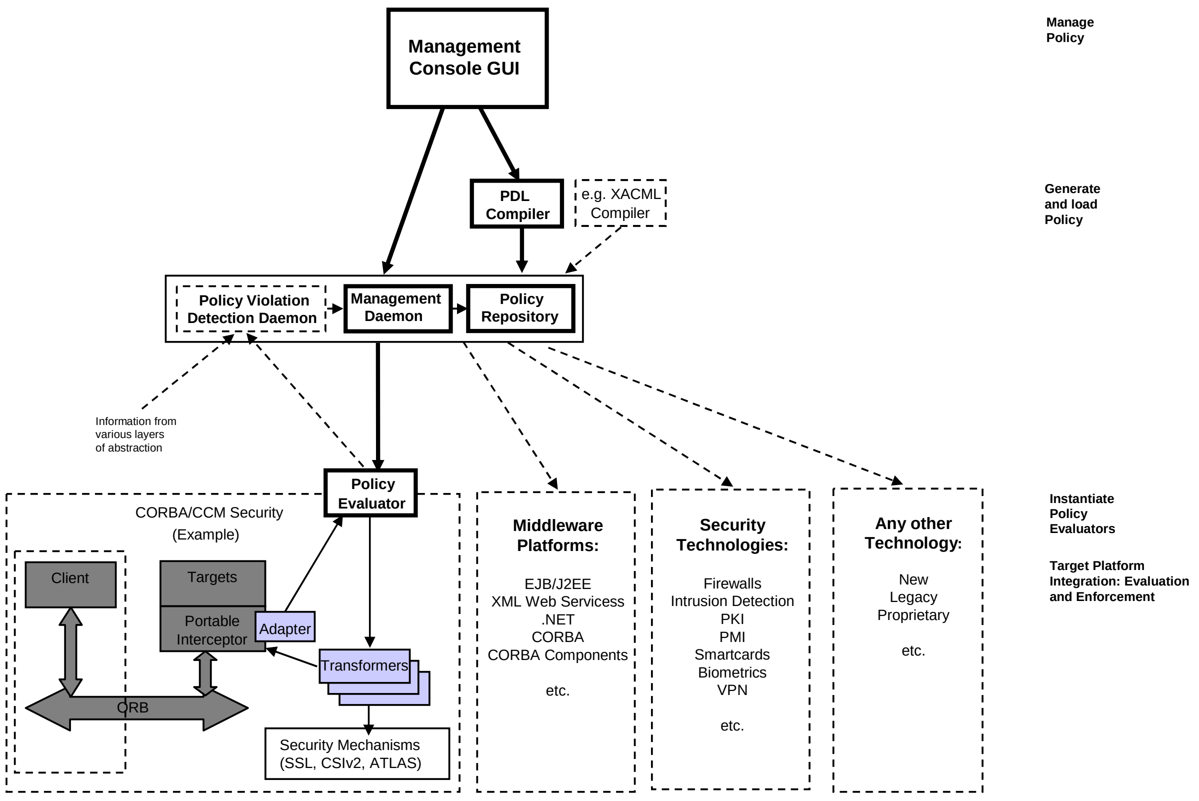
OpenPMF Architecture

In this paper, we describe OpenPMF, our policy management framework for distributed IT systems. A non-commercial version of OpenPMF is available as open source software under the GNU Public License at www.openpmf.org. There is also a commercial version called OpenPMF Enterprise Edition, which does not have the GPL restrictions. Additional features can be purchased on demand, such as an IIOP domain boundary controller that integrates with OpenPMF, an LDAP plug-in and, in the near future, a management console for the central administration of security policies. A previous prototype of OpenPMF has been developed as part of a European Union R&D project (www.ist-coach.org) in cooperation with a number of large communications and IT infrastructure companies. At this time of writing, OpenPMF is extended and significantly refined as part of another European Union R&D project on air traffic management (AD4).

One of its main features is that a technology-independent, human-readable security policy is stored centrally, consistently, and flexibly. This allows easy administration, policy optimisation, and correctness verification. Also, both legacy and future distributed systems platforms and security technologies can be integrated with OpenPMF.

OpenPMF is based on a model driven software development (Model Driven Architecture, MDA) approach in which the application logic is specified and maintained undistorted by underlying technologies and can therefore be migrated to new technologies with less effort. The crucial idea behind MDA is that the application logic has been modelled in the stable PIM and can therefore be migrated from one technology to another (future) middleware platform with acceptable effort. The MDA process consists of three basic steps, the Platform-Independent Model (PIM) application specification in UML, the conversion of the PIM into a Platform-Specific Model (PSM), and the application code generation. We argue in this paper that the concepts of model driven software development can be applied to security, and that this yields a number of advantages.

The diagram below shows an architectural overview of the OpenPMF framework. At start-up time, the technology-neutral policy is loaded into the policy repository. It is then obtained by the different systems, servers or applications and transformed into an efficient internal representation optimised for the evaluation of abstract attributes obtained from the underlying security technology and platform. At runtime, each incoming invocation triggers the evaluation process, after which the resulting decision is enforced on the particular underlying platform. This approach is inspired by the model-driven concept in that the abstract model is specified undistorted by underlying technologies. The current version is triggered by invocations arriving at the target side. OpenPMF is managed through the management daemon and the management GUI. In addition, the policy violation detection daemon collects relevant information from various layers of the underlying IT infrastructure and detects intrusions.



The areas marked with solid lines in figure 1 have been implemented in Java and C++, i.e. the CCM/CORBA mappings so far (e.g. MICO, Qedo, JacORB). The light grey boxes are platform specific, while the white boxes are platform independent. The dark grey boxes in the diagram show the exemplary infrastructure.

In the following, the main conceptual OpenPMF components are described.

Policy Definition Language

The policy is expressed using our human-readable, technology-independent Policy Definition Language (PDL), which supports different security models (analogous to PIM). It uses concepts of the Principal Calculus (Abadi et al) which theorises about principals and two different privilege delegation relations. Although PDL comprises a rich set of features, the actual policy that can be enforced on the platform is limited by the features supported by the underlying technologies. In OpenPMF, the evaluation process starts at the top of the policy and works its way down until a rule matches.

PDL supports rules are expressed in terms of requests and replies (i.e. invokers, intermediaries, actions, targets etc.). All PDL features can be arbitrarily combined, and the language can be flexibly extended (because the MOF repository is itself model-based) to incorporate new features: Firstly, wildcards, multiple sets, several (arbitrary) actions, sets of clients/targets, groups and roles, and hierarchical nesting are supported.

One of the most advanced features of PDL is its extended support for delegation which is particularly important in complex distributed systems because typically parts of the system delegate some of their work to other parts of the system. PDL supports two delegation modes: weak delegation (keyword “quotes”), which is useful if the intermediate and the target trust each other; and strong delegation (keyword “speaksfor”) if there is no sufficient trust between intermediate and target.

This is a brief example of a PDL policy file that controls access to a bank account application:

```
policy /OS [*, *] {

       // Admin allowed to write policy, bank server allowed to obtain policy
       policy /OS/Bank [/OS/Bank/Admin, /OS/Bank/Server] {

       // Simple rule
       (initiator.name == /OS/Director)&(operation.name == create)
              &(target.type ==  IDL:Bank:1.0) : allow;

       // All clients in group /OS/Accounting are allowed to open the account
       (initiator.group == /OS/Accounting)&(operation.name == open)
              &(target.type ==  IDL:Bank:1.0) : allow;

       // List of operations
         (initiator.group ==/OS/Accounting )&(operation.name == {deposit, balance})
              &(target.type ==  IDL:Account:1.0) : allow;

       // Again a simple rule
         (initiator.name == /OS/Director)&(operation.name == withdraw)
              &(target.type ==  IDL:Account:1.0) : allow;

       // Strong delegation
        (client.speaksfor.name == /OS/Director) &
              (initiator.group == /OS/Accounting)&(operation.name == withdraw)
              &(target.type ==  IDL:Account:1.0) : allow;
  };
};
```

In the near future, OpenPMF will support model-based policy definition on a higher level of abstraction. The security modelling front-end will draw information from the application and the underlying platforms, so that security policy definitions are much easier to formulate. This also gives a higher assurance for the correctness of the security policies.

Policy Repository

The PDL policy is fed into the MOF based Policy Repository. The benefits of using open MOF standard are extensibility, flexibility, XMI policy interchange, and automatic generation of the repository code and interfaces/descriptors. The technology-independent security policy compares to the PIM of the MDA.

Storing a technology-independent security policy centralized (ideally for the whole organisation) has several advantages: policy consistency, easier policy optimisation, redundancy detection, easy mapping to different (and future) technologies and technology versions.

The content of the Policy Repository can be managed through the Management Daemon. Moreover, the information stored in the Policy Repository is used by the Policy Violation Daemon to compare information from the underlying IT infrastructure with the policy.

Evaluator

When the application that is protected by OpenPMF is started, an efficient tree representation of the policy (with branches of different length) is instantiated based on the information obtained from the Policy Repository.

At runtime, the Policy Evaluator is called by the so-called Adapter, a platform specific piece of code that is interposed in the invocation path and intercepts all invocations. The Adapter triggers the entire policy evaluation process and enforces the result by blocking or granting the invocation.

The Policy Evaluator is a technology-unspecific interpreter for security rules that makes security policy decisions at runtime based on abstract attributes. Each time an invocation arrives at the target side, OpenPMF evaluates the policy by iterating through the tree. Storing the policy in a tree structure is more flexible than the traditional access control lists (ACL) or access control matrices (ACM). For example, it is not possible to capture delegation with traditional ACLs or ACMs, while our policy tree can include separate nodes for the initiator and the intermediate (because different branches can have a different number of levels from the root to the leaf) and therefore supports the “speaksfor” and “quotes” delegation modes mentioned above. Another weakness of traditional ACLs and ACMs is that all subjects and objects, respectively, are of the same uniform type and have to be compared using the same comparison function. Our approach allows different nodes to be of a different type, and different compare operations for each node. This increases the flexibility of the policy. In summary, while the security model is effectively hard-coded in traditional ACLs/ACMs, our tree-based approach can represent more flexible security models.

Transformers

OpenPMF is integrated with the underlying technology through so-called Transformers as follows: For each node in the tree the Policy Evaluator calls the Transformer, which compares the security attribute obtained from the underlying platform and security technology with the policy entry. If there is a match, then it iterates to the next node in the tree. If the Policy Evaluator reaches a leaf of the tree and there is a match, then the function(s) in the action list are executed.

Transformers have several purposes: Firstly, they obtain the security attributes used during policy evaluation from the underlying platform and security technology. Secondly, they translate technology-specific security information into technology-independent security attributes. Thirdly, Transformers can apply additional mappings on a higher level of abstraction, such as identity-to-role mappings or clustering into application domains. Finally, Transformers provide the Policy Evaluator with the required comparison functionality (equality, greater-than, etc.) used to compare the particular obtained attribute with the entry in the policy.

In each node of the policy tree, a Transformer chain is invoked to obtain and transform the relevant attributes. Each Transformer in the chain only carries out one particular attribute mapping (e.g. for obtaining an identity, for mapping it to an abstract name, and for mapping that abstract name to a role or group), which facilitates code reuse (e.g. if the underlying security mechanism changes, only the bottom transformers need to be replaced).

Transformers have to be hand-coded by platform specialists (once per platform, not once per application) and need to be integrated manually with the underlying technologies. The current OpenPMF version includes Transformers for CORBA/CORBA Security 1.x, CCM, and CSIv2. OpenPMF is also integrated (although not through Transformers) with a Public Key Infrastructure (PKI), a Privilege Management Infrastructure (PMI) based on the OMG Authorization Token Layer Acquisition Service (ATLAS), directory services based on OpenLDAP for storing user data and an IIOP Domain Boundary Controller (DBC).

In the near future, OpenPMF will be integrated with several other distributed systems technologies (e.g. J2EE, XML Web Services, Web servers, .NET Remoting) and security mechanisms (Kerberos, SAML); a tool that translates between XACML and PDL to allow the integration of upcoming XML Web Services security products; constraints expressed in the Object Constraint Language (OCL), fined grained information filtering, Mandatory Access Control (MAC), more advanced role-based access control models, and a new approach to the “owner” concept.

An important feature of OpenPMF is that Transformers and the Adapter have to be developed only once per platform, not for each application (corresponding to the MDA PSM and code generation steps). The fact that OpenPMF is available as Open Source means that users can integrate their (proprietary) IT infrastructure with the framework themselves. To provide production-grade product support, we offer high-quality, reliable, and cost-effective support contracts for OpenPMF. But there is no explicit vendor lock-in compared with proprietary software.

Policy Manager

The Policy Manager daemon is responsible for the management of the security policies and the evaluators. For example it sends policy update notifications to the evaluators, checks the states of the policies evaluators and the resources protected by them, and receives notifications, for example indications of policy violations. The Policy Manager is normally controlled by a graphical user interface.

Policy Violation Detection

The purpose of the prototypical Policy Violation Detection daemon is to collect and analyse information about activities in the underlying IT infrastructure. This information is on various technical layers of abstraction (e.g. application, middleware, network), and is not always easily accessible. The Policy Violation Detection daemon analyses this information and compares it with the policy in the Policy Repository. If a violation is detected, an alarm can be raised in the management console, or a custom action such as shutting down a port on a firewall can be automatically executed.

Conclusion

In OpenPMF, a technology-independent security policy is stored in a centralized, standardised policy repository for a number of distributed applications, platforms, and security technologies. The policy is expressed using a technology independent policy definition language. OpenPMF, which is available in Java and C++ as Open Source Software, is then integrated with the underlying technologies through well-specified interfaces.

This approach has a several advantages: First, security policies across many different platforms and security technologies can be integrated and stored centrally in one place, which ensures the policy consistency and optimisation across multiple platforms and security technologies. Secondly, the technology-independent policy language is easier to administer than the intricate details of the underlying technologies, and also makes the validation of the semantic correctness of the technology-independent policy feasible. From a more theoretical perspective (and related to the CSIv2 mapping) it is an advantage of OpenPMF that the same security theory based on the Principal Calculus is used from the abstract security policy language (PDL) down to the line-level protocol (CSIv2).

In summary, our OpenPMF policy management framework shows that the concepts of model driven software development (such as the OMG Model Driven Architecture) can be successfully applied to security. OpenPMF reduces the effort for the development of secure distributed applications, and helps to achieve integrated, consistent and correct security policies. As a result, OpenPMF minimises the total cost of security administration and increases security by reducing policy complexity, policy redundancy, policy inconsistency, and policy maintenance.

OpenPMF is available as open source software under the GNU Public License at www.openpmf.org. There is also a commercial version called OpenPMF Enterprise Edition, which does not have the GPL restrictions. Additional features, such as an IIOP domain boundary controller and an LDAP plug-in, are available on demand.

Using an Open Source licensing model for a security policy umbrella framework like OpenPMF has a number of advantages. Firstly, it is easier for the user to integrate a heterogeneous IT infrastructure, Secondly, it is easier to adapt the framework to new requirements. Thirdly, there is no vendor lock-in. Lastly, the correctness of the framework implementation can be examined by the user.