Policy Mediation for Multi-Enterprise Environments

P. Galiasso O. Bremer J. Hale * S. Shenoi
Center for Information Security
Department of Computer Science, Keplinger Hall
University of Tulsa, Tulsa, Oklahoma 74104, USA

D. Ferraiolo V. Hu
National Institute of Standards and Technology
Gaithersburg, Maryland 20899, USA

Abstract

Existing software infrastructures and middleware provide uniform security services across heterogeneous information networks. However, few, if any, tools exist that support access control policy management for and between large enterprise information networks. Insiders often exploit gaps in policies to mount devastating attacks. This paper presents a Policy Machine and Policy Mediation Architecture for coordinating diverse policies in large information networks. The language-based approach adopted by each of these technologies permits local and global access control policy validation with static analysis and other formal techniques. Together, the Policy Machine and Policy Mediation Architecture comprise an effective system for closing policy gaps in multi-enterprise environments.

1. Introduction

Secure interoperability is the overarching requirement for cooperative computing in multi-enterprise environments. Underlying software infrastructures have been designed to provide common security services across heterogeneous information systems and platforms [10,16,18,21]. Middleware facilities for secure communication, authentication, auditing and access control offer developers common mechanisms for application-level integration of low-level security functionality, but similar support is lacking for high-level security policy management and coordination.

Unfortunately, most attacks do not hinge on exploits for flawed communication protocols or sophistiacted cryptanalysis techniques. Rather, many attacks – commonly perpetrated by insiders – take advantage of gaps in enterprise security policies [7,22,24]. Not only are such attacks more pervasive, but they are also more destructive. Insiders, for instance, have wider access to sensitive resources, deeper knowledge of internal systems and greater opportunity to carry out their plans. This problem is exacerbated in multi-enterprise environments. Yet few, if any, tools exist to systematically coordinate and evaluate security policies for and between large enterprises. Ironically, the greatest potential for abuse is met with the least amount of analysis and prevention.

This paper describes a software architecture designed to foil policy gap attacks in federated systems, i.e., systems lacking centralized management. The proposed architecture employs a Policy Machine (PM) for universal policy expression and a Policy Mediation Architecture (PMA) for coordination and validation of diverse policies and models. The resulting solution helps administrators define strategies for achieving global policy consistency and resolving policy conflicts.

2. Policy Machine

The Policy Machine (PM) provides an elegant solution for reconciling heterogeneous access control models and policies in information enclaves. The PM constructs access control policies from a relatively small set of atomic properties completely expressed with mappings on three domains – *Objects*, *User Sets*, and *Operations*. Mappings and constraints are enforced with a fixed collection of databases and processing units.

2.1. Policy modeling

The PM is composed of a security policy language – the General Policy Language (GPL) – and an access control mechanism – the Policy Engine (PE) – that, together,

*To whom correspondence should be addressed (email: *johnhale@utulsa.edu*).

Research supported by MPO Contracts MDA904-96-1-0114, MDA904-96-1-0115 and MDA904-98-C-A900.

express and enforce heterogeneous access control policies. This includes support for popular access control policies such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Separation of Duty (SoD), One-Directional Information Flow, Workflow, Chinese Wall and N-Person Control. The PM is also flexible enough to model *ad hoc* policies used in industry, government and open Internet environments.

Internally, the PM maintains relations over the User Set, Operation and Object domains. (These relationships are themselves sensitive and so must be protected by PM policies that control the use of PM operations.) The PM abstraction eliminates the need for specialized access control mechanisms. For instance, the PM can simultaneously manage DAC and MAC by distinguishing between subject classes for users and administrators.

The PM in Figure 1 builds on the reference monitor concept. It is defined in terms of three access control processes and five authorization databases. The PM's core is a subject-object mediation function that allows active entities (subjects) to access passive entities (objects), based on subject/object attributes and prevailing authorization constraints.

A relevant set of security policies is created through the use of the General Policy Language (GPL). The GPL defines various elements and relationships within PM databases checked for property compliance. These include: (i) User Set Relations Database, (ii) Object/User Set/Operation Relations Database, (iii) Object/User Set/Operation Constraints Database, (iv) User Set Activation Constraints Database, and (v) History-Based Constraints Database. When sufficiently populated, the PM databases, collectively called the Authorization Database, embody a PM policy.

2.2. Policy Engine

The PE is a general security mechanism that can be configured to enforce any access control policy. The PE handles access requests of the form $<$ User Identity, [User Set], Operation, Object $>$, computing a Boolean output that determines whether access is granted or denied. The PE regulates access to target objects with respect to the policies specified using the GPL. The PE performs four basic functions to enforce these policies: User Set Activation, Subject-Object Mediation, Static Constraint, and History Filtration.

User Set Activation is triggered by access requests. It regulates the creation and assignment of attributes to subjects. In response, a new subject is created with Identity, Operation and Object attributes specified in a request. An Active User Set (AUS) is also constructed from the User Set Relations Database and the User Set Activation Constraint Database, which identify and constrain authorized user set relations, respectively.

Subject-Object Mediation constrains access based on privileges stored in the Object/User Set/Operation Relations Database. This function takes the subject created by User Set Activation as input and compares its attributes with privileges, i.e., (User Set, Operation) pairs, associated with the requested object. Authorization is given under two conditions: (i) There exists an element of the subject's AUS along with the requested operation that matches a (User Set, Operation) pair in the object's attribute list. (ii) No further Subject-Object Mediation checks are required under a different PM.

The Static Constraint function ensures that constraints stored in the Object/User Set/Operation Constraints Database are consistent with the relations stored in the User Set Relations and Object/User Set/Operation Relations Databases. Constraints stored in these databases are checked whenever a User Set, User Set/Operation, Operation/Object, Object/User Set/Operation or Object/User Set relation is created. These constraints can take the form of exclusive or inclusive relations. In addition, there exists a potential for inconsistencies between newly created static constraints and relations in the User Set or the Object/User Set/Operation databases. Therefore, an initial consistency check must be performed whenever new constraint relations are created.

From an implementation perspective, each of the databases used by these functions can be placed in a single schema. However, from a policy point of view, each is treated as a separate and independent database since it is used by different processes.

The History Filtration function is activated by an authorization event message passed from the Subject-Object Mediation process. It evaluates the relevance authorization events with respect to history-based policies stored in the History-Based Relations Database. History-based policies have two components: event and response. A Subject-Mediation event matching a list of events stored in the History-Based Relations Database triggers the response associated with the event in the database. Responses either update the User Set Activation Constraints Database or the Object/User Set/Operation Relations Database. A consistency check on the Object/User Set/Operation Constraints Databases when response components are initially created ensures that updates do not compromise their integrity.

The PM's most significant feature is that it provides a common representation for access control models and policies. Its inherent flexibility enables the expression of models and policies from practically any domain. Moreover, the PM is a foundation for negotiating security policies between enterprises.

Although the GPL and PM are capable of expressing any

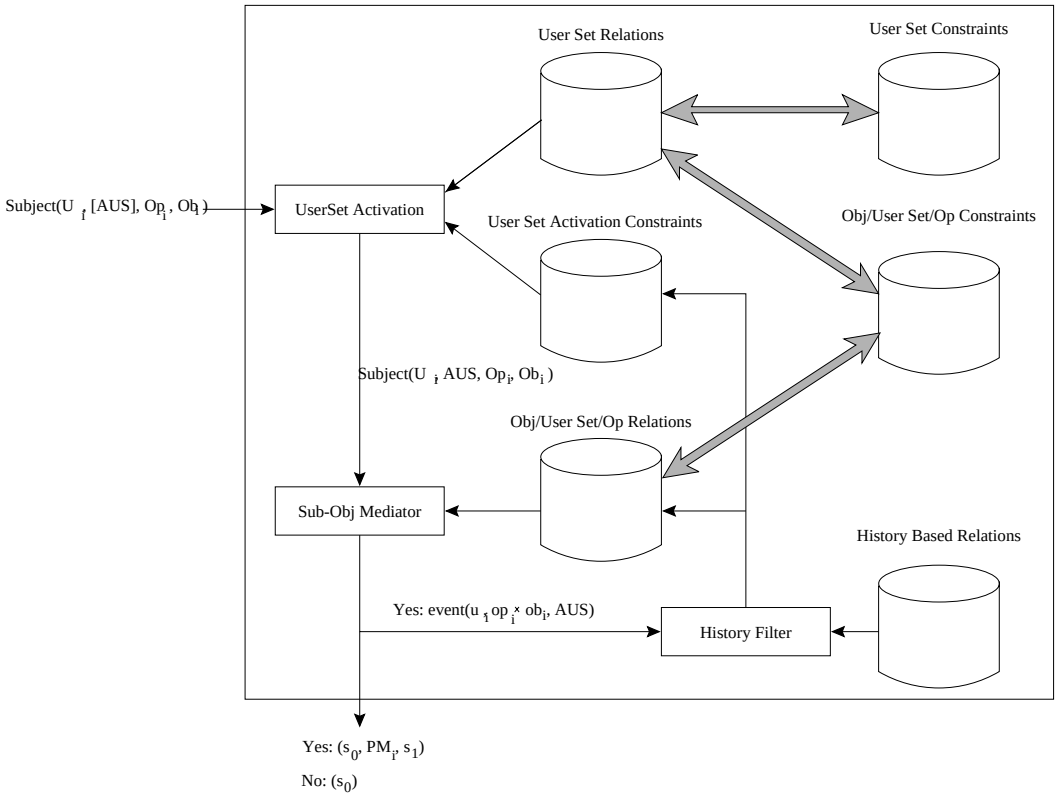


Figure 1. Policy Machine.

access control policy, not all combinational policies can be created within a single PM. For example, one-directional information flow policies, such as the DoD's Multilevel Security (MLS) policy and the Biba Integrity policy, cannot be composed into one PM with a role-based access control policy. However, individual PMs can be linked (i.e., the Boolean output of one PM is the input to another) to protect an object under multiple policies that cannot be collapsed into one. Thus, sequential and nested policy composition is possible under this design. While not all compositions represent viable enterprise policies, e.g., establishing policy precedence in many cases is critical, the ability to link PMs provides the flexibility to express complex, yet coherent enterprise security policies.

3. Policy mediation

Security policy coordination in distributed systems is achieved using a special Policy Mediation Architecture that coordinates security policy negotiation with the help of mediators. The architecture relies on JDBC and CORBA to provide a standard API for accessing heterogeneous information resources and to enable cross-platform application-level integration, respectively [10,16]. Each enterprise houses its own policy mediator. The policy mediator is a trusted piece of software that contains (i) an abstract model of the local information system, (ii) a Policy Machine that embodies its local prevailing policies, (iii) a Metapolicy Engine that lets security administrators specify policy conflicts, and (iv) policy and metapolicy rules.

3.1. Software architecture

While PMs enforce local authorization policies for single enterprises, policy mediators coordinate policies in multi-enterprise environments, facilitating secure interoperability. These mediators share policy, metapolicy and query metadata to negotiate common policies for individual access requests. Communication between mediators must be secure: Integrity and confidentiality are essential properties of the architecture. A Mediation Engine (ME) in each mediator transforms remote access requests (which can be fragments from a multi-enterprise query) into requests augmented with information received from other mediators. The augmented requests are then passed to the resident PM for authorization.

Mediators may export local policy and query information along distinct channels (Figure 2) to help negotiate an authorization policy for multi-enterprise access requests. E.g.,

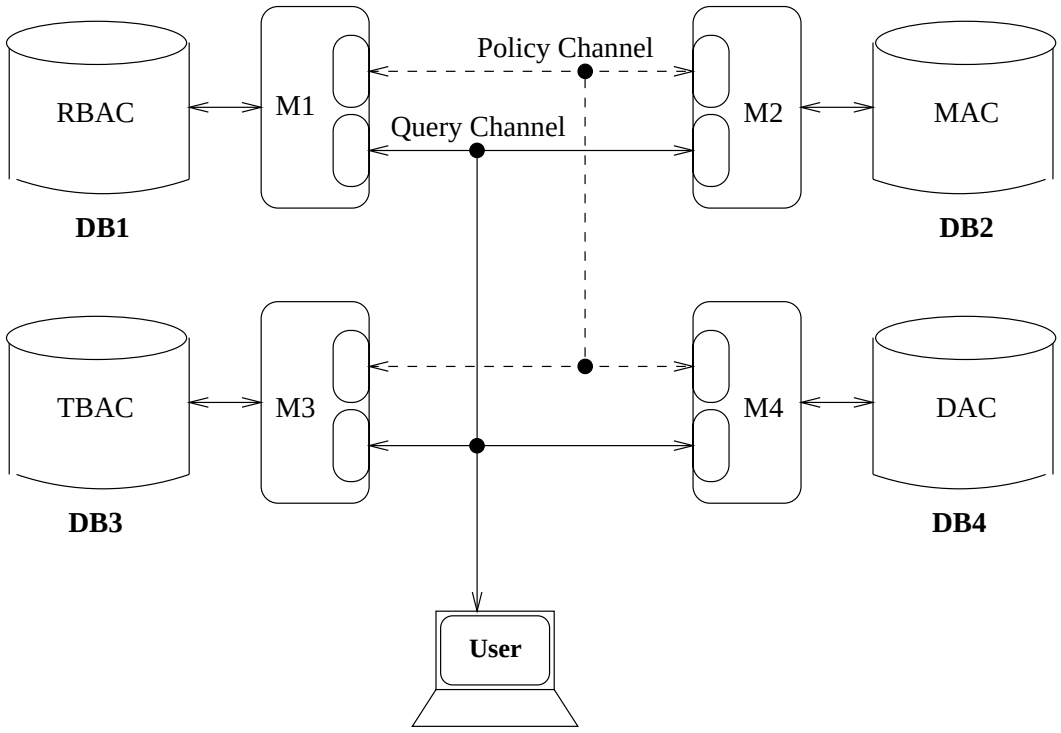


Figure 2. Policy Mediation Architecture.

it may be relevant that an enterprise allows universal access to $< Name, Position >$ relations, while another another hides $< Name, Salary >$ information, yet openly posts $< Position, Salary >$ data. By sharing policy and query information, a global authorization policy can be derived locally that eliminates such data aggregation vulnerabilities. Conversely, a derived policy may be more permissive if shared local policies give a subject higher privileges.

3.2. Metapolicy Engine

The Metapolicy Engine accepts information exported by other mediators and applies metapolicy rules to derive policies for multi-enterprise access requests. Metapolicy rules are expressed in a language for making policy statements about authorization policies. These rules provide a systematic means of altering a local policy to extend or retract trust boundaries.

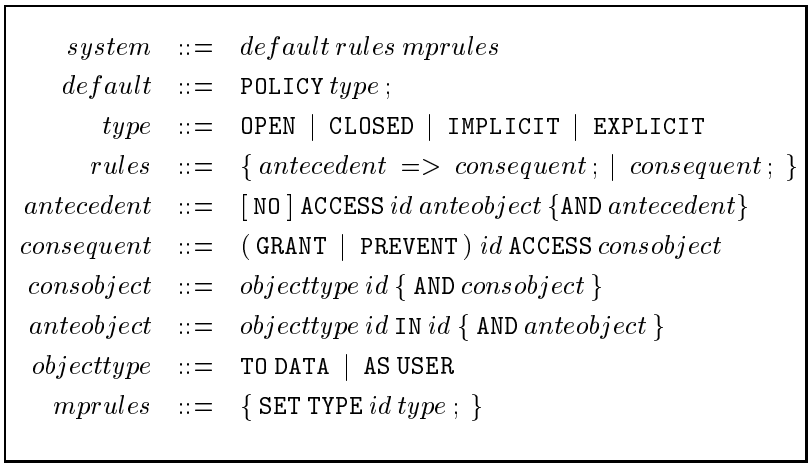
An EBNF grammar (Figure 3) defines the metapolicy specification language. Rules for "bending" security policies are based on information held and shared by mediators. Policies are bent at each site by mediators according to these prevailing metapolicy rules. E.g., the rule in Figure 4, as applied to a university's information resources, extends the local policy by granting access to an information resource element under a conjunctive conditional.

The rule antecedent specifies a user bound to variable *x* as an engineer at WCM with access to PMA resources at NIST. The rule consequent grants users (who are bound to the antecedent conditions) access to ProjectX information held at the university. Antecedents are satisfied by information held locally or imported from remote mediators.

In some cases, metapolicy rules produce conflicts, where one or more rules grant access to resource while others prevent access. For example, a conflict occurs in Figure 5 when an engineer at WCM has access to PMA resources at NIST and DOD. The first rule grants access to the ProjectX at the university while the second one prevents it.

In general, a conflict occurs when either two rules produce contradictory access decisions (explicit conflict) or when no rule can be evaluated to produce an access decision (implicit conflict). Conflict resolution policies are specified with special syntax in the metapolicy grammar (Figure 3). Metapolicy keywords `OPEN`, `EXPLICIT`, `IMPLICIT`, and `CLOSED` indicate distinct conflict resolution policies.

`OPEN` grants access to resources in the face of any conflict, i.e., when no access decision can be made or when contradictory metapolicy evaluations are returned. `IMPLICIT` specifies a policy of granting access when a conflict is implicit, and denying access for explicit conflicts. `EXPLICIT` applies the converse strategy, granting access for explicit conflicts and denying access when metapolicy rules apply. Finally, `CLOSED` always denies access in the presence of any conflict, explicit or implicit.



$$
\begin{array}{rcl}
system & ::= & default\ rules\ mprules \\
default & ::= & \texttt{POLICY}\ type\ ; \\
type & ::= & \texttt{OPEN} \mid \texttt{CLOSED} \mid \texttt{IMPLICIT} \mid \texttt{EXPLICIT} \\
rules & ::= & \{\ antecedent\ \texttt{=>}\ consequent\ ; \mid consequent\ ;\ \} \\
antecedent & ::= & [\ \texttt{NO}\ ]\ \texttt{ACCESS}\ id\ anteobject\ \{\texttt{AND}\ antecedent\} \\
consequent & ::= & (\ \texttt{GRANT} \mid \texttt{PREVENT}\ )\ id\ \texttt{ACCESS}\ consobject \\
consobject & ::= & objecttype\ id\ \{\ \texttt{AND}\ consobject\ \} \\
anteobject & ::= & objecttype\ id\ \texttt{IN}\ id\ \{\ \texttt{AND}\ anteobject\ \} \\
objecttype & ::= & \texttt{TO DATA} \mid \texttt{AS USER} \\
mprules & ::= & \{\ \texttt{SET TYPE}\ id\ type\ ;\ \}
\end{array}
$$

Figure 3. EBNF grammar.

```
ACCESS x AS USER "Engineer" IN "WCM" AND
ACCESS x TO DATA "Policy Machine" IN "NIST"
=> GRANT x ACCESS TO DATA "ProjectX";
```

Figure 4. Metapolicy rule.

Conflict resolution keywords can be applied at multiple levels of resource granularity, specifying conflict resolution strategies for individual relations, databases or whole enterprises. Furthermore, the metapolicy specification language requires a default conflict resolution strategy, ensuring that a prevailing strategy exists for every enterprise resource. Finer-grained conflict resolution specifications override coarser-grained specifications.

The default conflict resolution strategy in Figure 6 is specified as `CLOSED`, except for the ProjectX resource, which is associated with an `EXPLICIT` strategy. Therefore, engineers at WCM having access to RFC867 at NIST can access ProjectX at the university regardless of whether or not they have access to CFP412 at DoD.

Objects engaging `EXPLICIT` conflict resolution strategies are not affected by preventing rules, only by granting rules. Similarly, objects with `IMPLICIT` resolution strategies are not affected by granting rules and not by preventing rules. A metapolicy compiler issues warnings about inconsistent metapolicy rules, including those detected as ineffectual in the presence of prevailing resolution specifications.

Mediators bend local authorization policies by transforming metapolicy and query fragments into updates on local PM relations. Furthermore, metapolicy specifications can seamlessly incorporate results from remote PMs. This permits the expression of combinatorial and nested access control policies. Using metapolicy specifications to connect a network of PMs yields a flexible framework for multi-enterprise policy management and coordination. Finally, the formal syntax and semantics of the metapolicy specification language provide an opportunity for static analysis to detect and resolve policy and metapolicy conflicts.

4. Related work

This section compares the proposed architecture with other access control solutions for heterogeneous policy environments and large federated systems. Managing heterogeneity at the policy level is particularly challenging. De Capitini di Vimercati and Samarati [6] have proposed an authorization model that addresses these problems for tightly-coupled federations. Their solution guarantees authorization autonomy by supporting decentralized access control between local sites. But it relies on a central authority for managing federated schema and objects. Our solution is more attractive in that it focuses on federations where no central authority is possible.

The well-known Argos system [14] unifies heterogeneous access control models in an open distributed environment by permitting the configuration of various identity-based authorization policies. Argos is one of the largest efforts to achieve a multipolicy authorization model and implementation for object-oriented databases. Argos also introduced the notion of domains for subject behavior and object protection rights. Domains generate classes of protection rights and behavior, materializing basic access

```
ACCESS x AS USER "Engineer" IN "WCM" AND
ACCESS x TO DATA "Policy Machine" IN "NIST"
=> GRANT x ACCESS TO DATA "ProjectX";

ACCESS x AS USER "Engineer" IN "WCM" AND
ACCESS x TO DATA "Policy Machine" IN "DOD"
=> PREVENT x ACCESS TO DATA "ProjectX";
```

Figure 5. Metapolicy conflict.

```
DEFAULT POLICY "CLOSED"

ACCESS x AS USER "Engineer" IN "WCM" AND
ACCESS x TO DATA "RFC867" IN "NIST"
=> GRANT x ACCESS TO DATA "ProjectX";

ACCESS x AS USER "Engineer" IN "WCM" AND
ACCESS x TO DATA "CFP412" IN "DoD"
=> PREVENT x ACCESS TO DATA "ProjectX";

SET "ProjectX" TYPE "EXPLICIT"
```

Figure 6. Conflict resolution specifications.

types (e.g., read and write) from complex object systems. Identity-based authorization models are prevalent, but our proposed Policy Machine advances Argos by permitting the expression of radically different policy models.

A common theme in authorization model research is the separation of access control mechanisms from the policies themselves. The principal benefit of this approach is that policies can be defined (and redefined) with a specification language without requiring any modification to underlying mechanisms. Unfortunately, existing security policy specification languages are not general enough to express all access control policies. While certain policy specification languages [9,12,13] capture a range of access control features and concepts (e.g., postive/negative authorizations, user/groups/roles separation of duty), they are limited, unlike the proposed approach, by the imagination of their designers.

Most policy specification languages, e.g., [17,19,20], are tied to existing mechanisms. These languages and policy models are limited by the scope of the security attributes and mediation functions in a particular access control mechanism. For example, Ahn and Sandhu [2] propose a language for expressing several separation of duty (SoD) policies based on the role-based access control (RBAC) model [20]. Although the language is rich enough to express role-based SoD policies, it is not suited to expressing one-directional or workflow policies.

Flexible security architectures have been proposed that allow security rules to be defined by external software components [1,20,23]. This permits the altering of policies by redefining subject-object security attributes and rule sets within an external subject-object mediation function. Because policy flexibility is limited to the reference mediation function, established security policies expressed by imposing constraints on administrative operations or through applying a historical context are left outside the scope of control.

Schemes supporting multipolicy access control are becoming increasingly relevant [3,4,11]. The architecture in [4] employs flexible mechanisms and mediators [25] for tunable access control policies. Candan *et al.* [5] used the HERMES system for the secure mediation of heterogeneous databases. Their work promotes the "principle of cautious cooperation" to enforce local and global security policies while providing widely integrated services. Dawson *et al.* [8] describe a mediator architecture for reconciling heterogeneous data and multilevel secure (MLS) lattices in federated databases. However, the proposed research is unique in that policy heterogeneity is addressed at the model level, i.e., the integration of RBAC, MAC and DAC policies.

The "metapolicy concept" was introduced in [11] and applied to policy negotiation via the "multipolicy ma-

chine" in [3]. Kuhnhauser and von Kopp Ostrowski [15] used metapolicies to construct a formal framework supporting multiple access control in loosely-coupled federations. Their framework provides support for application-specific policy development, co-existence and re-use in open environments. Metapolicies in the framework are realized through the construction of cooperation and conflict matrices – policies themselves are the objects in these matrices. While their approach does not deal directly with heterogeneous policy models, the implementation in [15] uses "custodians," which are similar to the mediators used in our architecture.

5. Conclusions

Policy gap attacks, commonly orchestrated by insiders, represent a pervasive and destructive threat to multi-enterprise information networks. Such attacks are often enabled by a lack of coordination between large enterprises. Tools for policy analysis, management and negotiation are needed to close gaps and prevent malicious users from exploiting weak policies. The Policy Machine (PM) described in this paper defines a common representation and software architecture for modeling and enforcing heterogeneous access control policies. A Policy Mediation Architecture (PMA) deploys multiple PMs across federated information networks and engages a metapolicy scheme for connecting enterprise policies and resolving conflicts. Moreover, the language-based approach adopted by each provides an opportunity for validating local and global access control policies via static analysis and other formal techniques. Together, the Policy Machine and Policy Mediation Architecture comprise an elegant solution for coordinating policies in multi-enterprise environments.

References

[1] M. Abrams *et al.* (1990) A generalized framework for access control: An informal description. *Proc. 13th National Computer Security Conference*, 135-143.

[2] G. Ahn and R. Sandhu (1999) The RSL99 language for role-based separation of duty constraints. *Proc. 4th ACM Workshop on Role-Based Access Control*.

[3] D. Bell (1994) Modeling the multipolicy machine. *Proc. New Security Paradigms Workshop*, 2-9.

[4] E. Bertino *et al.* (1996) Supporting multiple access control policies in database systems. *Proc. IEEE Symposium on Research in Security and Privacy*, 94-109.

[5] K. Candan *et al.* (1996) Secure mediated databases. *Proc. 12th International Conference on Data Engineering*, 28-37.

[6] S. De Capitani di Vimercati and P. Samarati (1997) Authorization specification and enforcement in federated database systems. *Journal of Computer Security*, **5(2)**, 155-188.

[7] D. Denning (1999) *Information Warfare and Security.* Addison-Wesley, Reading, Massachusetts.

[8] S. Dawson *et al.* (1998) Secure interoperation of heterogeneous systems: A mediator-based approach. *Proc. 14th IFIP TC-11 International Conference on Information Security*.

[9] V. Gligor *et al.* (1998) On the formal definition of separation-of-duty policies and their composition. *Proc. IEEE Symposium on Research in Security and Privacy*.

[10] G. Hamilton, R. Cattell and M. Fisher (1997) *JDBC Database Access With Java: A Tutorial and Annotated Reference*. Addison-Wesley, New York.

[11] H. Hosmer (1993) The multipolicy paradigm for trusted systems. *Proc. New Security Paradigms Workshop*, 19-32.

[12] S. Jajodia *et al.* (1997) A logical language for expressing authorizations. *Proc. IEEE Symposium on Research in Security and Privacy*, 31-42.

[13] S. Jajodia *et al.* (1997) A unified framework for enforcing multiple access control policies. *Proc. ACM SIGMOD Conference*, 474-485.

[14] D. Jonscher and K. Dittrich (1995) Argos – A configurable access control system for interoperable environments, in *Database Security, IX: Status and Prospects* (eds. Spooner, D. *et al.*), Chapman-Hall, London, 43-60.

[15] W. Kuhnhauser and M. von Kopp Ostrowski (1995) A framework to support multiple security policies. *Proc. 7th Annual Canadian Computer Security Symposium*.

[16] T. J. Mowbray and R. Zahavi (1995) *The Essential CORBA: Systems Integration Using Distributed Objects*. John Wiley, New York.

[17] National Computer Security Center (NCSC) (1987) *A Guide to Understanding Discretionary Access Control in Trusted Systems*. Report NSCD-TG-003 Version 1.

[18] W. Rosenberry, D. Kenney and G. Fisher (1993) *Understanding DCE*. O'Reilly and Associates, Inc., Sebastopal, California.

[19] R. Sandhu (1993) Lattice-based access control models. *IEEE Computer*, **26(11)**, 9-19.

[20] R. Sandhu *et al.* (1996) Role-based access control models. *IEEE Computer*, **29(2)**, 38-47.

[21] O. Saydjari *et al.* (1993) *Synergy: A Distributed, Microkernel-Based Security Architecture*, Version 1.0. Technical Report, National Security Agency, Fort Meade, Maryland.

[22] F. Schneider (Ed.) (1999) *Trust in Cyberspace.* National Academy Press, Washington D.C.

[23] R. Spencer *et al.* (1999) *The Flask Security Architecture: System Support for Diverse Security Policies*, `www.cs.utah.edu/fluz/flask`.

[24] The White House (2000) *Defending America's Cyberspace: National Plan for Information Systems Protection (Version 1.0)*, Washington, D.C.

[25] G. Wiederhold (1992) Mediators in the architecture of future information systems: A new approach. *IEEE Computer*, **25(3)**, 38-49.