RECENT ADVANCES IN ACCESS CONTROL MODELS

Sushil Jajodia
Center for Secure Information Systems
George Mason University, Fairfax VA 22030-4444, USA
jajodia@gmu.edu

Duminda Wijesekera
Center for Secure Information Systems
George Mason University, Fairfax VA 22030-4444, USA
dwijesek@gmu.edu

Abstract Advances in application areas bring new dimensions to access control needs. This paper discusses several access control models that have been recently proposed to address these emerging needs including models that provide policy-neutral flexible access control and their efficient implementations; models that incorporate richer semantics for access control in emerging Internet applications, such as adding provisions; and models for XML documents. We also discuss the recent work on policy algebras and subject identity issues in secure federations.

Keywords: Access control, security, security policy

1. INTRODUCTION

Traditionally, access control plays an integral part in overall system security. Over the years, many different access control models have been developed, and discretionary and mandatory access control models have received considerable attention. Discretionary access control is based on having subjects, objects, and operations as primitives and policies that grant access permissions of the form (o, s, a), where subject o is allowed to execute operation a on object o. Mandatory access control is based on having clearance levels for subjects and classification labels for objects as primitives and policies that grant accesses to subjects whose clearance levels are higher than those of the objects they access. These models have been used in the commercial and military domains, and

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35587-0_24
M. S. Olivier et al. (eds.), *Database and Application Security XV*

implemented in operating systems, database management systems and object-oriented systems.

Advances in application areas bring new dimensions to access control models. The needs to support multiple access control policies in one security domain, Internet-based transactions, cooperating coalitions, and workflow systems have brought new challenges to access control. In response, new access control models are being proposed to address these emerging needs.

Section 2 addresses problems posed by and solutions proposed to solve the problems in the area of multiple access control policies in one security domain. Section 3 discusses the concept of *provisional authorizations* that was introduced to meet the needs in the area of Internet commerce. In response to a user request, these applications requires decisions that go beyond the traditional *yes/no* answers. Section 4 addresses problems faced in merging multiple security policies of collaborating organizations. Section 5 addresses issues related to managing identities of subjects in such cases. Section 6 discusses role-based access control(RBAC), access control problems posed by extensible markup language (XML) documents, and workflow systems. Finally, Section 7 concludes the paper.

2. NEED FOR FLEXIBLE ACCESS CONTROL MODELS

Large numbers of access control models proposed over the years [Dobson and McDermid, 1989] have been developed with a number of pre-defined policies in mind and thereby have introduced a sense of *inflexibility*. Two alternatives accommodate more than one access control model simultaneously. The first is to have more than one access control mechanism running at the same time, one for each policy. The second is to make access control an application responsibility. The first alternative calls for every application to be closely bound to its access control module, which decreases their portability. The second alternative requires all applications to enforce a consistent access control. Additionally, the responsibility of enforcing access control is vested in applications; it will not impose the same rigorous standards of verification and testing imposed on system code.

Consequently, both alternatives are undesirable. This can be seen by considering a number of access control policies that have been used over the years [Castano et al., 1994]. A popular policy is the *closed world* policy, where accesses that cannot be derived from thoses explicitly authorized are prohibited. A rarely used alternative is the *open world* policy, where accesses that are not explicitly denied are permitted. Some policies include explicit prohibitions in terms of negative authorizations. This, coupled with generalizations, specializations of these policies to structures such as subject and object hierarchies [Bruggemann,

1992, Rabitti et al., 1991] yield numerous combinations. Hence, custom creation of policy enforcement mechanisms or passing of these complications to applications is practically infeasible.

One of the solutions for this problem has been to develop flexible authorization models [Jajodia et al., 1997, Jajodia et al., 2001b], where the flexibility comes from having an access control model that does not depend on any policies or meta policies, but is capable of imposing any of them specifiable in their syntax. One of the main advantages of this approach is that access control can now reside within the system, but yet be able to impose application specific policies. Given that there is a need for flexible access control models, the following requirements would be desirable:

Expressability: It must be possible to model existing policies, such as *denials take precedence*, and be able to model policies of emerging applications, such as provisions and obligations (to be discussed shortly).

Decoupling Policies from Mechanisms: The primary need for flexibility is to obtain policy-independent frameworks. Hence, policies expressible in such frameworks must be enforceable using generic enforcement mechanisms.

Conflict Resolution: Having a flexible framework may invite conflicting policies and, consequently, the framework must be able to facilitate their resolution.

Efficiency: Due to the high frequency of requests coming to access control systems, their processing must be fast. Thus, efficient and simple mechanisms to allow or deny access requests are crucial.

Next we describe one such flexible framework for access control and show how it meets these requirements.

2.1. FAF - A Flexible Authorization Framework

The Flexible Authorization Framework (FAF) [Jajodia et al., 2001b], is a logic-based framework to specify authorizations as rules in a stratified rule base. The FAF architecture has four modules as shown in Figure 1.

The *propagation module* contains basic structures such as authorization subjects and object hierarchies (for example, directory structures) and a finite set of rules is used to derive authorizations stemming from structural properties, referred to as *propagation policies*. Propagation policies specify how access permissions are to be propagated through subject and object hierarchies, and FAF allows the system security officer (SSO) to write them. This freedom may result in overspecification, implying that the system must allow and deny the same access request simultaneously. Therefore, the *conflict resolution module*

implements *conflict resolution policies* to eliminate such inconsistencies and these policies are specifiable by the SSO. By applying decision policies, which is again a set of rules written by the SSO, a decision will be made either to grant or deny every authorization request. This stage has a meta-policy denying all permissions that cannot be specifically derived using the given set of rules. The last stage consists of checking for integrity constraints, where all authorizations that violate integrity constraints will be denied. This component lives outside the scope of the rule base and is an external module used by FAF.

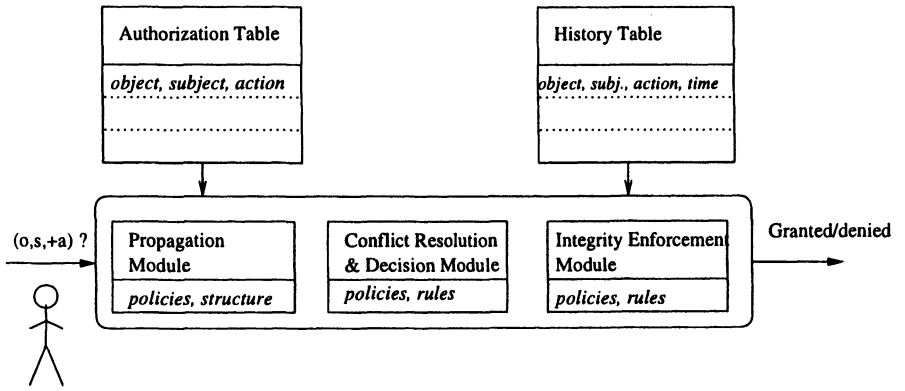

Figure 1. FAF System Architecture

FAF rules are written using a number of predicates, such as `cando`, `do`, and `dercando`. Their semantics are as follows:

1 A ternary predicate `cando(o,s,a)`, representing grantable or deniable requests (depending on the sign associated with the action), where o, s, and a are object, subject, and a signed action term, respectively.

2 A ternary predicate `dercando(o,s,a)`, with the same arguments as `cando`. The predicate `dercando` represents authorizations derived by the system using logical rules of inference (modus ponens plus rules for stratified negation [Apt et al., 1988]).

3 A ternary predicate `do`, with the same arguments as `cando`, representing the access control decisions made by FAF.

4 A 5-ary predicate `done(o, s, r, a, t)`, meaning subject `s` with role `r` active has executed action `a` on object `o` at time `t`.

5 Two 4-ary predicate symbols $\texttt{over}_{\texttt{AO}}$ and $\texttt{over}_{\texttt{AS}}$. $\texttt{over}_{\texttt{AO}}$ takes as arguments two object terms, a subject term and a signed action term. $\texttt{over}_{\texttt{AS}}$ takes as arguments a subject term, an object term, another subject

term, and a signed action term. They are needed in the definition of some of the overriding policies.

6 A propositional symbol `error` indicating violation of an integrity constraint. It is a rule with an `error` head that must not have a body that is satisfiable.

An example policy governing the electronic trading is given by the following FAF rules:

$$
\begin{aligned}
\texttt{cando}(item, s, +buy) &\leftarrow \texttt{in}(item, Goods, ASH), \\
&\quad\ \texttt{in}(s, Buyers, ASH). \\
\texttt{cando}(item, s, +sell) &\leftarrow \texttt{in}(item, Goods, ASH), \\
&\quad\ \texttt{in}(s, Sellers, ASH). \\
\texttt{dercando}(item, s, +a) &\leftarrow \texttt{cando}(item, s, +a). \\
\texttt{do}(item, s, +a) &\leftarrow \texttt{dercando}(item, s, +a) \\
\texttt{do}(item, s, -a) &\leftarrow \neg\texttt{do}(item, s, +a). \\
\texttt{error} &\leftarrow \texttt{do}(item, s, +buy), \texttt{do}(item, s, +sell)
\end{aligned}
$$

The predicate in(x, y, hierarchy name) is used to specify properties of subject and object hierarchies AOH and ASH, respectively. ASH consist of two directories, `Buyers` and `Sellers`. The object hierarchy AOH has one class, `Goods`. Rules whose heads are $\texttt{dercando}(o, s, a)$ literals are derivations of authorizations. Thus, the first two rules state that a subject `s` is allowed to buy or sell if it is in the appropriate directory. The next stage in the sequence is conflict resolution and ensuring the completeness of access control decisions, and they are stated in two rules with $\texttt{do}(o, s, a)$ heads. Hence, all derived positive permissions are allowed, and all actions for which positive permissions cannot be derived are denied. The last step is *integrity checking* and is given by rules with an `error` head. The integrity rule says that no subject is allowed to buy and sell the same item.

With respect to our requirements, logically, expressability of FAF is limited to that of stratified logic programs, but many policies, such as closed world and open world policies, denials-take-precedence policy, and Chinese Wall policy, are expressable in FAF [Jajodia et al., 2001b]. In addition, to capture the evolving nature of policies and, consequently, their impact on access control, FAF has been enhanced to add and remove rules [Jajodia et al., 2001b], and to remove already granted access permissions [Wijesekera et al., 2001]. What would be interesting is to find ways to incorporate error-handling capabilities into the logical framework of FAF so that constraints specified by important classes of application-level policies such those used in role-based policies [Ahn and Sandhu, 2000] can be resolved inside the FAF rule enforcement engine.

For the decoupling of policies from their enforcement mechanisms, FAF has no built-in policies except for the meta-policy of refusing access that cannot be derived by any combination of rules. Notice that first it is necessary, as otherwise the system would be incomplete, and second, the stance taken by this meta-policy is arbitrary and can be changed to an open meta-policy by changing the rule about $\neg \mathrm{do}(s, o, -a)$. Conflict resolution in FAF is totally left to the SSO and hence is satisfactorily modularized.

Finally, the efficiency of FAF depends upon the execution of the stratified rule base. But FAF [Jajodia et al., 2001b] materializes rules and authorization tables, enhancing its efficiency.

3. PROVISIONAL AUTHORIZATIONS

Traditional access control uses the model that *a user makes an access request of a system in some context, and the system either authorizes the access request or denies it.* Models vary in what types of accesses are being considered (e.g., read, write, query, execute a function), who makes the request (e.g., user, member of a group of individuals), what objects are being accessed (e.g., file, directory), and what constitutes the context of the request (e.g., role). However, today's rapidly expanding environments, such as electronic commerce, make such models that authorize or deny a request overly simplistic and less accommodative. In practice, it is not unusual that decisions regarding accesses depend on specific actions to be performed before the decision is taken and on the guarantee that certain other actions will be taken after the access. Since the two sets of actions are conceptually different and require different management techniques, we distinguish the first and second set of actions as *provisions* and *obligations*, respectively.

An example is electronically purchasing goods for credit. The purchaser will be able to purchase an item online subject to the proviso that she enters her credit card, she is found to be *credit worthy*, and she agrees to pay back the loan. Notice that the original access decision is based on three provisions: entering the credit card, finding the purchaser to be credit worthy, and signing the purchase agreement. Once the provisions are satisfied and the purchase is completed, the purchaser is obligated to pay back the loan according to the credit agreement (notice that a credit card user's signature attests to this at the time of purchasing). To capture such applications, *provisional* authorization models have been proposed [Jajodia et al., 2001a, Kudo and Hada, 2000]. Two example systems that provide provisional authorizations are as follows.

3.1. Extending FAF with Provisions

Figure 2 shows the architecture of the provisional authorization system proposed in [Jajodia et al., 2001a]. When users submit an authorization request,

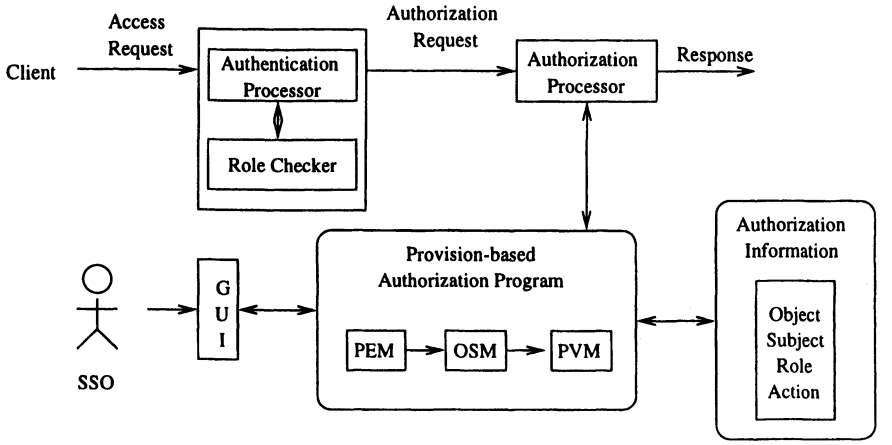

Figure 2. Provision-based Authorization Architecture

the system invokes its authentication and role-checking modules that verify if the user is who she claims to be and whether she is allowed to assume requested roles. Then, the access request is passed on to the provision evaluation module to find the *weakest* conditions under which the requested access can be honored. Then, the weakest condition under which the access may be granted is passed to an *order specification* module that yields a set of ordering constraints on the actions involved. For instance, the ordering constraints may require that the name and address be filled in before the social security number. Then, the ordering constraints are handed off to a *provision verification module* to check if any conditions were previously fulfilled by the requester and, if so, simplifies the condition (and ordering constraints) and waits for reduced conditions to be fulfilled by the requester before final authorization. Formally referred to as $\mathsf{pASL}_{\mathcal{L}}$, the syntax is enhanced to be of the following form (where $Head \leftarrow Body$ is a FAF rule and ϕ is a predicate external to FAF):

$$\phi : Head \leftarrow Body$$

The following set of rules models a $\mathsf{pASL}_{\mathcal{L}}$ specification for provisional accesses. An online store allows its customer to purchase by registering and further allows the customer to upgrade her registration to a *preferred customer.* These two policies are stated in the first two rules. Next two rules state that the purchase price of an item is \$100 for a non-preferred cutomer and \$80 for a preferred customer. Thus, a customer has the choice of remaining in the non-preferred category and paying \$100 or registering as a preferred customer and paying \$80 per item. Further suppose there is a one-time cost of \$10 to register as a non-preferred customer and to pay a \$20 fee for upgrading. Then it

is preferable to buy as a non-preferred customer for a one-time-only purchase, but to become a preferred customer for multiple purchases. This is so because the cost of the one-time purchase is \$80 after paying a one-time fee of \$30, as opposed to paying \$100 after paying a registration fee of \$10. $\text{pASL}_{\mathcal{L}}$ provides this computation for its customers.

```
register(s, customer): cando(item, s, +buy) ←
                           in(contract, Contracts).
upGrade(s, prefCust):  dercando(item, s, +buy) ←
                           cando(item, s, +buy).
payFees(s, $100):      do(item, s, +buy) ←
                           cando(item, s, +buy).
payFees(s, $80):       do(item, s, +buy) ←
                           dercando(item, s, +buy).
```

3.2. Provisional Authorizations for XML

Provisional authorizations have been used in XML [Bray, 1998] documents [Kudo and Hada, 2000]. Specifically, the work of Kudo et al. tailors the general idea of provisions to XML documents as described below:

- Limits objects to those given by `XPath` [XPathP, 1999] expressions.
- Ensures that provisions require the system to transcode XML documents, carried out by a *transcoding* module. Provisions are specified as actions to this module and are limited to `logging` (requests), `verifying`(digital signatures of requestors), `encrypting` (contents to be given to requestor), `transforming` (the contents of documents to fit release constraints stated by the access control module), and `read, write, create,` and `delete` new data structures to fit the request.
- Limits the access propagation policy in XML documents to propagating permissions upwards and downwards, and preventing propagation along hierarchies [Jajodia et al., 2001b].
- Limits conflict resolution policies to denials take precedence, permissions take precedence, and nothing takes precedence [Jajodia et al., 2001b].
- Provides a syntax referred to as XACL.

It is noteworthy that from the access requestor's perpective, the security transcoding *filters* the fields of the object according to the policies applicable within the access control system, without the requestor's knowledge, to those occurring in subject-switching algorithms proposed in [Yang et al., 2001].

4. ALGEBRA OF POLICIES

Another area that has advanced in recent years is that of policy algebras for access control [Bonatti et al., 2000, Wijesekera and Jajodia, 2001]. Due to mergers and aquisitions in the commercial sector and joint missions in the military sector, many security-sensitive organizations need to merge their security policies. Such policies exist, but they are based on different irreconcilable models and implemented in incompatible languages. Consequently, an algebra of policies goes a long way in providing conceptual coherence among these policies at a higher abstract level, thereby providing a basis for comparison and determining areas of contrast. Details of two recent approaches are given in Sections 4.1 and 4.2.

4.1. The Modular Approach of Bonatti et al.

Bonatti et al. [Bonatti et al., 2000] develop a comprehensive framework for composing access control policies, where an access control policy is a set of ground terms over an alphabet for (subject, object, action) terms. Thus, it is a set-based approach influenced by logic programming. BNF definitions of policies are given below, where **id** is a terminal and T is a template. Here, C consists of arithmatic constraints constructed from $<, \leq, =, \geq$, and $>$ signs. R represents the recursive closure under a set of Horn clauses:

$$p := \mathbf{id}, \mid p + p \mid p \& p \mid p - p \mid p^C \mid o(p,p,p) \mid p * R \mid T(p) \mid (p)$$

$+$ and $\&$ are, respectively, union and intersection operators, permitting accesses that are allowed under either or both components, respectively. The difference operator $(-)$ allows accesses permitted under the first component but not under the second. p^C restricts the scope of policy p to those accesses satisfying the constraint C, and $p * R$ represents recursive closure of policy p under the rule set R. $o(p_1, p_2, p_3)$ is syntactic sugar for $(p_1 - p_2) + (p_1 \& p_2)$ and represents policy overriding. T is a template and $T(p)$ represents its instantiation with p.

The paper further shows how to translate policy expressions into logic programs by creating a labeling mechanism for the syntax tree of the expression, which is referred to as the *canonical translation*. Although the modular approach does not explicitly support negative authorizations, the paper shows how to model a policy such as denials take precedence as $p^+ - p^-$, where p^+ and p^- contain positive and negative authorization terms of p, respectively.

4.2. Propositional Policy Algebra

The algebra proposed in [Wijesekera and Jajodia, 2001] is a propositional version consisting of a syntax to view policies as abstract symbols and their

semantics as authorization state transformers. Here, an authorization state is a collection of (subject, object, access set) triples and a set of propositions satisfied by them. BNF definitions of policy expressions of [Wijesekera and Jajodia, 2001] are given below, where p is a non-terminal, p_a is a terminal taken from a countable collection $\{p_i : i \geq 1\}$ atomic policies, and ϕ is a propositional symbol:

$$p := p_a \mid p \cup p \mid p \cap p \mid p - p \mid p; p \neg p \mid \odot p \mid p^* \mid (\phi p) \mid (p \| \phi) \mid \max(p) \mid \min(p)$$

The disjunction operator ($\cup$) permits accesses that are allowed under either of its components, while conjunction operator ($\cap$) allows only those that are permitted by both components. The difference operator ($-$) permits accesses that are allowed under the first component, but not under the second. The sequence operator ($;$) permits accesses that are allowed as a consequence of its second component after the first is applied. The negation operator ($\neg$) changes all positive authorizations to negative ones and vice versa. Applying the invalidate operator ($\odot$) to a policy removes all authorizations granted under that policy. The closure operator (*) allows accesses permitted under repeated application of policies and is an extension of the composition operator. The reason for recursion is to allow accesses that are permitted by some rules in a rule base. Provisions (:) are as described in Section 3. The scoping operator ($\|$) allows only those authorizations that meet the scoping restrictions. In case both positive and negative permissions are granted, `max` and `min` operators resolve conflicts by selecting the positive and negative permissions, respectively.

Semantically, policies allow specified subjects to execute desired actions over given objects. Accordingly, policies are interpreted as abstract transformers of (subject, object, action set) triples. The paper also provides some syntactic rules to simplify policy expressions and indicates how they can be used to determine completeness, consistency, unambiguity, and semantic equivalence of merged policies.

5. SUBJECT IDENTITY IN SECURE FEDERATIONS

Consider the problem of providing authorized accesses to a secure coalition such as one launched by allied forces against Afghanistan, Iraq, or Bosnia. Federated databases and systems are suitable in providing support for such federations, where each participant wants to contribute to the common mission but maintain a high degree of security and operational autonomy. In providing access control to such systems, there are two alternatives. One is to create individual subjects in each cooperating system. The other alternative is to allow every participant *anonymous* access to some data. None of these alternatives are completely satisfactory, as the former places extra burden on the users because they have to manage many accounts and are responsible for interpreting, translating, and collecting information that they may not have the knowledge to

manage. The second alternative is infeasible as this is an *all-or-nothing* solution that limits fine-grained access control offered by components.

A viable alternative to this problem is *subject switching*, where a federated layer interprets data supplied by the components and switches the identity of the subjects to those of subjects in participating components. The advantage of this approach is that the federated subject then has a lesser burden in using the system with a finer grain of access control, and the disadvantage of the system is that federated users may not get exactly what they request, as these may be limited by agreements and policies existing between the components and the federation. An access control model and algorithms to do the best under a given set of policies are discussed in [Yang et al., 2001].

6. OTHER RECENT ADVANCES

Role-based access control (RBAC) is an active area in access control, where the accesses to objects are dependent on the roles played by the subjects at the time the accesses are granted. It is an alternative to discretionary and mandatory access control, and has several advantages, such as in organizations with a stable structure, but a dynamic workforce accesses can remain relatively static and bound only to roles. Prevention of fraud and misuse are also enforceable by imposing *separation of duty* constraints. Recent advances in this area have been a proposal for standards [Ferraiolo et al., 2001], role constraint languages RCL [Ahn and Sandhu, 2000], applying RBAC to the World Wide Web [Sandhu, 1996], and applying RBAC to workflow systems [Bertino et al., 1999].

Another topic of recent interest has been in the area of securing XML documents [Damiani et al., 2000]. Their security depends on three aspects: authenticity, integrity, and confidentiality. Confidentiality depends on access control policies and mechanisms. Some issues that are being addressed in access control of XML are specifications that take the structure of XML documents into account, fine-grained access control of document components and fields, and access control based on credentials.

Workflow systems also continue to pose considerable challenges for access control [Atluri, 2001], where accesses must synchronize with workflows. Several recent attempts apply RBAC to workflows [Atluri et al., 2001, Bertino et al., 1999], where constraints posed by workflows need to be reconciled with those of RBAC models.

7. GOOD NEWS

In recent years, researchers and developers have devoted a great deal of energy on topics such as firewalls, incorporation of encryption on communication protocols, and intrusion detection systems. However, there is a growing realization that while these are critical to building secure systems, they do not

provide all the answers and, consequently, the focus is shifting toward host and application security. We need to accelerate our research and work with vendors to develop the practical uses of our solutions.

Acknowledgement

This work was partially supported by the National Science Foundation under the grant CCR-0113515.

References

[Ahn and Sandhu, 2000] Ahn, G.-J. and Sandhu, R. (2000). Role-based authorization constraints specification. *ACM Transactions on Information and Systems Security*, 3(4).

[Apt et al., 1988] Apt, K., Blair, H., and Walker, A. (1988). Towards a theory of declarative knowledge. In Minker, J., editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, San Mateo.

[Atluri, 2001] Atluri, V. (2001). Security for workflow systems. *Information Security Technical Report*, 6(2):59–68.

[Atluri et al., 2001] Atluri, V., Chun, S., and Mazzoleni, P. (2001). A chinese wall security model for workflow systems. In *ACM Conference on Computer and Communications Security*.

[Bertino et al., 1999] Bertino, E., Ferrari, E., and Atluri, V. (1999). An approach for the specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and Systems Security*.

[Bonatti et al., 2000] Bonatti, P., di Vimercati, S. D. C., and Samarati, P. (2000). A modular approach to composing access control policies. In *Proc. 7th ACM Conf. on Communications and Security*, pages 164–173.

[Bray, 1998] Bray, T. (1998). *Extensible Markup Language (XML) 1.0*. World Wide Web Consortium (W3C), http://www.w3.org/TR/REC-xml, 1.0 edition.

[Bruggemann, 1992] Bruggemann, H. (1992). Rights in an object-oreinted environment. In Landwehr, C. and Jajodia, S., editors, *Database Security V: Status and Prospects*, pages 99–115. North Holland.

[Castano et al., 1994] Castano, S., Fugini, M., and Samarati, P. (1994). *Database Security*. Addison-Wesley.

[Damiani et al., 2000] Damiani, E., di Vimercati, S. D. C., Paraboschi, S., and Samarati, P. (2000). Design and implementation of an access control processor for xml documents. *Computer Networks, vol. 33, no. 1-6, 2000, pp. 59-75*, 33(1-6):59–75.

[Dobson and McDermid, 1989] Dobson, J. and McDermid, J. (1989). A framework for expressing models of security policy. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 229–239.

[Ferraiolo et al., 2001] Ferraiolo, D. F., Sandhu, R., Gavrilla, S., Kuhn, D. R., and Chandramouli, R. (2001). A proposed standard for role-based access control. *ACM Transactions on Information and Systems Security*, 4(3).

[Jajodia et al., 2001a] Jajodia, S., Kudo, M., and Subrahmanian, V. S. (2001a). Provisional authorizations. In Gosh, A., editor, *E-Commerce Security and Privacy*, pages 133–159. Kluwer Academic Press, Boston.

[Jajodia et al., 2001b] Jajodia, S., Samarati, P., Sapino, M. L., and Subrahmanian, V. S. (2001b). Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260.

[Jajodia et al., 1997] Jajodia, S., Samarati, P., and Subrahmanian, V. (1997). A logical language for expressing authorizations. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, CA.

[Kudo and Hada, 2000] Kudo, M. and Hada, S. (2000). Xml document security based on provisional authorizations. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 87–96.

[Rabitti et al., 1991] Rabitti, F., Bertino, E., Kim, W., and Woelk, W. (1991). A model of authorization for next-generation database systems. *ACM Transactions on Database Systems*, 16(1):89–131.

[Sandhu, 1996] Sandhu, R. (1996). Role hierarchies and constraints for lattice-based access control. In *Proceedings of the Europeon Symposium on Research in Computer Security*, pages 65–79.

[Wijesekera and Jajodia, 2001] Wijesekera, D. and Jajodia, S. (2001). Policy algebras for access control - the propositional case. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security*. to appear.

[Wijesekera et al., 2001] Wijesekera, D., Jajodia, S., Parisi-Presicce, F., and Hagestrom, A. (2001). Removing permissions in the flexible authorization framework. Submitted for publication.

[XPathP, 1999] XPathP (1999). *XML Path Language (XPath)*. World Wide Web Consortium (W3C), http://www.w3.org/TR/PR-xpath19991008, 1.0 edition.

[Yang et al., 2001] Yang, J., Wijesekera, J., and Jajodia, S. (2001). Subject switching algorithms for access control in federated databases. In *Proceedings of the Fifteenth Annual IFIP WG 11.3 Working Conference on Database and Applications Security*.