Logic-Based Access Control Policy Specification and Management

Vladimir Kolovski[1]

Department of Computer Science, University of Maryland, College Park, MD 20740, USA

Abstract. Recently there has been a great amount of attention to access control languages that can cover large, open, distributed and heterogeneous environments like the Web. These languages aim to be flexible and extensible, with enough features to capture expressive and distributed security policies. However, with expressive languages such as XACML or WS-Policy, users have problems understanding the overall effects and consequences of their security policies. Even the task of checking that the policy will not result in leakage of permissions to an unintended or unauthorized principal is tedious and error-prone when done manually. As a result, there has been a great amount of research on logic-based policy management that provides analysis services to help find inconsistencies/differences between access control policies. This paper provides an overview of the existing approaches for security (access control) policy analysis. The survey covers both language proposals that have formal semantics and provide algorithms for policy analysis out of the box, and formalizations of already existing policy languages (WS-Policy, XACML, XrML, ODRL) that provide a formal semantics and analysis services previously unavailable for the particular language.

1 Introduction

With the widespread use of Web services, systems on the Web are becoming more connected and integrated. To protect the sensitive information that is often contained in these systems, there is an increased need for adequate security and privacy support. As a result, there has been a great amount of attention to access control policy languages for web services which accommodate large, open, distributed and heterogeneous environments like the Web. These languages aim to be flexible and extensible, with enough features to capture expressive and distributed security policies.

With these expressive policy languages, a significant problem is that users have difficulty understanding the overall effects and consequences of their security policies. Even arguably the most important feature in access control – checking that the access control policy will not result in the leakage of permissions to an unintended or unauthorized principal, i.e., *safety* - has become difficult, if not impossible, to do manually. For example, incomplete security policies might unintentionally give access to an intruder. How can a security administrator be

certain that her policy covers all possible corner cases? Even if the administrator does discover a bug in the policy, and fixes it accordingly, the ramifications of that fix (policy change) are difficult to analyze. It is possible that due to subtle interactions of some other policy rules with the fix, the change opened other security holes. Considering that most recent language proposals support distributed policies, support for conflict resolution and distributed evaluation, verifying that policies are safe has became an overwhelming task for security administrators. In addition, the lack of formal semantics for most industry proposals in this space has caused even more ambiguity over the particular meanings of policies written in different languages.

To address the above concerns, there has been a great amount of research into security policy analysis [38, 37, 2, 32, 16, 24, 52, 45, 16, 7]. In the research community there have been a number of analysis services proposed - the most common being verification of a policy against given safety properties. For example, as a part of a company-wide access policy, one could state 'Junior developers should never be allowed to sign expense reports' or 'at any given time, a user cannot be a member of more than two of the following three roles {JuniorDeveloper, SeniorDeveloper, AccountsClerk } '. Then, company security officers would use automated tools to *verify* their policy against these constraints. In the event bugs are discovered and the policy is changed, administrators would perform change analysis to ensure no new bugs were unintentionally introduces, using queries of the form 'Show me all requests involving Expense Reports that used to map to Deny but now are mapping to Permit'. The above mentioned services of verification and change analysis have been proposed as the building block of a useful policy analyzer tool ([16]). In addition to the above two, there are many other analysis services that have received some attention:

- **Policy Redundancy** - A policy P is called redundant if when removed from the policy set, the behavior of the access control system does not change. In other words, P either never applies or is always overridden by another policy higher up in the hierarchy.
- **Policy Incompatibility** - If for two policies P_1 and P_2, there cannot be a request s.t. both policies yield an access decision (both apply), then these policies are *incompatible*.
- **Policy Coherence** - Check whether a policy P would ever return a Permit (or Deny).
- **Policy Repair** - In the case when the policy does not satisfy the safety property, compute the smallest set of constraints that need to be added in order to fix it.
- **Coverage Checking** - Checks for a given description of a resource whether an access request exists that will be not be evaluated to a Permit or Deny.

This paper will provide a survey of logic-based policy analysis approaches. First, In Section 2 we provide a brief overview of logic-based policy specification languages, with focus on Semantic Web based languages. In Section 3 we provide a survey of logic based policy analysis approaches, which is the main

contribution of this paper. In Section 3 we discuss policy specification languages with analysis support. In Section 3.1 we discuss approaches that map existing policy languages to a formalism, which is then used to provide analysis services for those languages. We provide some focus on XACML, since because of its expressiveness it has attracted a great amount of attention from the security analysis research community.

2 Logic-Based Access Control Policy Specification

In this section I will discuss policy language proposals that are based on logic. These languages all benefit from having unambiguous semantics and well understood computational properties. Most of them are based on Datalog, so evaluating whether an access request satisfies the policy is done in PTIME. Some of the proposals [27] even materialize the unique model of the underlying datalog program, so there is no inference done at runtime.

One of the earliest attempts at a general, logic-based framework for expressing authorizations was made by Woo and Lam [51], who proposed the use of default logic to model authorization and control rules. Default logic is a very expressive formalism, so Woo and Lam propose to use a fragment of the logic that corresponds to stratified, extended logic programs, where the unique program model can be computer in polynomial time. Using this formalism, they provide a formalization of the Bell-LaPadula security model [12]. The authors do not discuss any analysis services.

A similar approach is Delegation Logic [34], where the authorization logic is limited such that tractable results are achieved. Delegation Logic combines the following features: it is based on logic programs, expresses delegation depth explicitly and supports a wide variety of complex principles (including but not limited to k-out-of-n threshold). In addition, Delegation Logic provides a concept of proof-of-compliance that is not entirely ad-hoc and is based on model-theoretic semantics. Delegation Logic combines these features and can be extended with non-monotonicity, negation and prioritized conflict handling. In their approach, there is no attention paid to analysis and verification of policies.

PeerAccess [50] is a framework for reasoning about authorization in open distributed systems. It supports a declarative description of the behavior of peers that selectively push and/or pull information from certain other peers. PeerAccess local knowledge bases encode the basic knowledge of each peer, its policies governing the release of each possible piece of information to other peers - in this sense of information release it is similar to PeerTrust [18]. PeerAccess proofs of authorization are verifiable and nonrepudiable, and their construction relies only on the local information possessed by peers and their parameterized behavior with respect to query answering, information push/pull, and information release policies.

In addition to PeerAccess and Delegation Logic, there are other policy languages that support delegation: Abadi et al[1], SecPal [4], Binder [14], SD3[29] ,RT [36] and Cassandra [41]. All but Abadi et al. use Datalog as basis for syntax

and semantics. SecPal is a simple (semantics consists of three deduction rules) yet very expressive language: it can express many common policy idioms using constraints, controlled delegation, recursive predicates and negated queries. Cassandra, RTC and SecPal are all based on stratified Datalog with constraints for higher expressiveness. A limitation of all of these languages is that they do not support classical negation.

Proof-carrying Authorization (PCA) and related distributed proof systems [3] are an authorization framework based on a higher-order logic where different domains in the system use different, less expressive, application-specific logics. The higher-order logic (AF logic) used to check the proofs is undecidable, though this problem is avoided by forcing clients to generate proofs on their own, using only a decidable subset of AF logic. Consequently, the authorizing servers task of proof-checking is reduced to a tractable type-checking problem - however this leads to large rate of increase of sizes of the client proof.

Jajodia et al. [26] have proposed a logical language for specification of authorizations that allows users to specify, together with the authorizations, the policy according to which access control decisions are to be made. Policies are expressed by means of rules which enforce derivation of authorizations, conflict resolution, access control, and integrity constraint checking. The Flexible Authorization Framework (FAF) [28] they propose corresponds to a quadratic time data complexity fragment of logic programming. This is accomplished by stratifying the authorization predicates in FAF. Also, the authors propose a materialization technique that allows for incremental updates of the policy model at run-time.

There is a body of work on dynamic policies, i.e., policies that change over time. First we will discuss approaches that focus on changing subjects, resources or authorizations, and then discuss approaches that have static policy bases, but allow for temporally dependent authorizations.

The framework of Harrison, Ruzzo and Ullman [22] is one of the earliest approaches that allows for changing number of subjects, roles, resources, and authorizations. The HRU model is very expressive; it could model most of the protection systems in use at that time when it was proposed. However, because of the expressiveness, there is no algorithm to decide if a given subject can eventually obtain an access privilege to a given object (it is undecidable).

Bertino et al [5] present a temporal extension of the RBAC model called TRBAC. TRBAC supports periodic role enabling and disabling—possibly with individual exceptions for particular users—and temporal dependencies among such actions, expressed by means of role triggers. Role trigger actions may be either immediately executed, or deferred by an explicitly specified amount of time. Enabling and disabling actions may be given a priority, which is used to solve conflicting actions. A formal semantics for the specification language is provided, and a polynomial safeness check is introduced to reject ambiguous or inconsistent specifications. They also present an implementation of TRBAC on top of a conventional DBMS.

2.1 Semantic Web-Based Policy languages

Recently there has been a great amount of attention to how Semantic Web technologies can be used in policy systems. In particular, there have been a number of proposals that show how to ground or express policies in a Semantic Web framework [49, 30, 31, 47].

Rei [30] is a policy specification language based on a combination of OWL-Lite, logic-like variables and rules. It allows users to develop declarative policies over domain specific ontologies in RDF and OWL. Rei allows policies to be specified as constraints over allowable and obligated actions on resources in the environment. A distinguishing feature of Rei is that it includes specifications for speech acts for remote policy management and policy analysis specifications like what-if analysis and use-case management.

The successor of Rei is Rein [31], which is a policy framework grounded in semantic web technologies that allows for different policy languages and supports heterogeneous policy systems. Rein provides an ontology for describing policy domains in a decentralized manner and provides a reasoning engine built on top of CWM , an N3 rules reasoner. Using Rein and CWM, the authors show how it is possible to develop domain and policy language specific security systems. Rein has been successfully used as a policy management system in the Policy Aware Web project [49], which in turn provides an architecture for scalable, discretionary, rule-based access control in open and distributed environments.

PeerTrust [18] also deals with discretionary access control on the web using semantic web technologies. It provides a mechanism for gaining access to secure information/services on the web by using semantic annotations, policies and automated trust negotiation. In PeerTrust, trust is established incrementally through an iterative process which involves gradually disclosing credentials and requests for credentials. PeerTrusts policy language for expressing access control policies is based on definite Horn clauses. A distingiushing feature of PeerTrust is that it expects both parties to exchange credentials in order to trust each other and assumes that policies are private, which is appropriate for critical resources such as military applications and e-commerce sites.

Finally, KaOS Policy and Domain Services [47] use ontology concepts encoded in OWL to build policies. These policies constrain allowable actions performed by actors which may be clients or agents. The KAoS Policy Service distinguishes between authorizations and obligations. The applicability of the policy is defined by a class of situations which definition can contain components specifying required history, state and currently undertaken action.

All of the above proposals address the challenges that a policy language should overcome to be usable in a massively open and distributed setting. Thus, they mostly focus on architectural, privacy and scalability issues for web policies. None of them discuss or propose any analysis services (except policy subsumption in KaOS). Additionally, most of the above policy languages use quite expressive logics so they can cover variegated use cases. It is an open question whether analysis services can be provided in a scalable manner for such expressive languages.

3 Logic-Based Policy Analysis

There has been research on security analysis for access control [38, 37] that uses the notions of *states* of policy systems and *transitions* (for example, adding a role, or changing a permission) that alter those states. Then, usually a set of queries is proposed that investigates the possible consequences of certain changes in the policy. Simple safety checking is an example of a query; it checks if there exists a reachable state in which a (presumably untrusted) principal has access to a resource. Although early results show that this type of safety analysis can easily lead to undecidability [22], there has been recent work that demonstrates a class of access control models and queries for which safety is decidable and efficient algorithms exist [38]. There are some limitations to the expressiveness of the models that are analyzed: the states are described using positive Datalog programs only.

Elisa Bertino et al [6] propose a formal framework for reasoning about different access control models. Their framework is logic-based and can capture discretionary, mandatory, and role-based access control models. Each instance of the proposed framework corresponds to a C-Datalog program, interpreted according to the stable model semantics. To demonstrate the expressiveness, the authors map the Bell and La Padula model [12] and NIST RBAC [44] to their framework. The authors also propose some dimensions along which access control models can be analyzed and compared. For example, they show that checking for structural subsumption/equivalence between different access control models is decidable, however access request equivalence is not.

Chomicki and Lobo [9] introduce a declarative policy description language $\mathcal{PDL}$, in which policies are described as sets of event-condition-action (ECA) rules. They provide a framework for detecting and resolving conflicts between the ECA rules and any action constraints. This is performed using a policy monitor, which in order to resolve conflicts chooses or ignores certain events, essentially preventing the ECA rule from activating and causing the conflict. The semantics of the ECA rules, and conflict detection and resolution are defined using logic programs. They also describe the architecture of a PDL-based policy server that is used to provide centralized administration of a switch in a communication network and show how it can be augmented to provide conflict resolution.

Dougherty et al. [15] present a model for formal analysis of access-control policies in their dynamic environments. In particular, they propose a new mathematical model of policies, their environments, and the interactions between them. Then the authors propose two core analysis services: a) goal reachability, which checks if there is some accessible state in the dynamic access model which satisfies some boolean expression over the policy facts and b) contextual policy containment, which intuitively asks if one policy is more permissive than another. These services are provided using a combination of relational reasoning and temporal reasoning. A unique feature of their model is that it separates the static policy from its dynamic environment. This separation allows for analysis services such as semantic differencing that can be applied to the static policy alone.

Lithium [20] is a language for reasoning about digital rights and is based on a fragment of first order logic. It is different from Datalog-based approaches since it allows real logical negation in the conclusion as well as in the premises of policy rules. To show that the Lithium is expressive enough, the authors gather a large collection of policies from different types of libraries and map them to their policy language. They also show how large fragments of XrML [21] and ODRL [42] can be translated in the language. They provide two types of analysis services:

- Given a set of policies ,a policy environment and an access request, does it follow that the access request is permitted by the policy set?
- Consistency checking of a policy set. Unlike [26], the language does not support conflict resolution mechanisms, so whenever both a permit and deny is returned by a policy, it is treated as an error.

In order to remain decidable, Lithium restricts recursion and cannot easily express delegation.

Moving to state-based approaches, Schaad et al. [45] examine the problem of verifying a policy that is subject to change coming from another policy. Using the Alloy [25] specification language and its model-checking facilities, they show how to specify an RBAC96-style model, ARBAC97-style extensions and a set of separation of duty properties. The authors do not present any implementation or evaluation results.

In [52] the authors present a model-checking algorithm which can be used to evaluate access control policies, and a tool which implements it. The evaluation includes not only assessing whether the policies give legitimate users enough permissions to reach their goals, but also checking whether the policies prevent intruders from reaching their malicious goals. Policies of the access control system and goals of agents are described in the access control description and specification language *RW* [19]. Their algorithm takes a policy description and a goal as input and performs two modes of checking. In the assessing mode, the algorithm searches for strategies consisting of reading and writing steps which allow the agents to achieve their goals no matter what states the system may be driven into during the execution of the strategies. In the intrusion detection mode, a weaker notion of strategy is used, reflecting the the willingness of intruders to guess the value of attributes which they cannot read.

There are also proposals for analyzing policies based on description [53] or modal logics [40]. Both of them provide a formalization of role based access control, and show how tableau-based decision methods can be used for consistency checking of policies, evaluating access requests and verifying policies against security properties. Zhao et al [53] present a formalization of RBAC based on the description logic ALCQ. They also show how RBAC policy constraints (separation of duty, role hierarchies) can be captured with this logic. Massacci [40] formalizes RBAC using multi modal logic and presents a decision method based on analytic tableaux. Because he is using tableau-based algorithms, he is able to provide services similar to ours: logical consequence, model generation and consistency checking of policies.

In [2], the authors propose a set of services under the name of policy ratification. In particular, they present algorithms for analysis tasks such as dominance, coverage and consistency check that can be performed independently of policy model and language and require little domain-specific knowledge. They present algorithms from constraint, linear, and logic programming disciplines to help perform ratification tasks. They provide an algorithm to efficiently assign priorities to the policies based on relative policy preferences indicated by policy administrators. Finally, they show how these algorithms have been integrated with a working policy system to provide feedback to a policy administrator regarding potential interactions of policies.

3.1 Applying Logic to Reason about Existing Languages

As mentioned in the previous section, the authors of Lithium have mapped large fragments of ODRL [42] and XrML [21] to their FOL-based language. In addition to giving ODRL formal semantics, the authors explored the practical problem of determining whether a set of ODRL statements implies a permission or prohibition. Using Lithium's semantics, they showed the problem is NP-hard. They also showed that by removing a component of ODRL whose meaning seems to be somewhat unclear, a tractable fragment of the language results. They prove that the fragment is tractable by creating a polynomial-time algorithm to determine if a set of ODRL statements implies an access decision. They achieve a similar result for XrML in [21]: propose a formal semantics, show that deciding access requests in the language is NP-hard, then propose an expressive fragment of the language for which deciding access requests is polynomial.

Kolovski et al. [33] present a formalization of the core WS-Policy model using OWL-DL. Using the OWL mapping, they show how off-the-shelf OWL reasoners can be used to act as policy processors. The authors a set of analysis services (policy comparison, consistency checking, disjointness/compatilibity, coherence) that go beyond what is usually offered. Additionally, the authors provide debugging support by leveraging research in explanation for inconsistencies in OWL-DL KBs. In a more recent work [48], the authors extend the WS-Policy coverage by coupling OWL with default logic constructs.

In [35] the authors present a first-order logic (FOL) semantics for SDSI [43]. Additionally they show how the FOL semantics can be easily extended to additional policy concepts and gives meaning to a larger class of access control and other policy analysis queries. The authors prove that the FOL semantics is equivalent to the string rewriting semantics used by SDSI designers, for all queries associated with the rewriting semantics. Using their semantics, they discover a few issues in the proof procedures for SPKI/SDSI defined in RDFC 2693 [8]. Finally, they compare SPKI/SDSI with $\mathrm{RT}_1{}^{\mathrm{C}}$, a Datalog-based language that is part of the RT policy framework.

Reasoning about XACML In this section I'll discuss approaches that formalize and analyze fragments of XACML [24, 52, 45, 16, 46, 7, 10]. The semantics

of XACML is expressed in terms of the functional language Haskell. Since most of the constructs have a declarative flavor, researchers have tried mapping them to a logical semantics. In particular, this is true of conflict resolution and policy combination methods, that are specified procedurally in the XACML specification while could also be expressed declaratively.

Hughes et al. [24] propose a framework for automated verification of access control policies based on relational First-Order Logic. They introduce a formal model for systematically specifying access to resources, and show that the access control policies in the XACML access control language can be translated to a simple form which partitions the input domain to four classes: permit, deny, error, and notapplicable. The authors show how to automatically verify policies using an existing automated analysis tool, Alloy [25]. Because using the first-order constructs of Alloy to model XACML policies is prohibitively expensive (in terms of performance), the authors use only the propositional constructs. However, it is unclear from their results whether it is feasible for larger policies. In addition, the results of policy analysis are an internal Alloy representation that can only be explored with Alloy's visualization tools, and cannot be queried or processed in more detail.

Bryans et al. [7] formalize XACML policies using a process algebra known as Communicating Sequential Processes (CSP [23]). This allows them to use a model checkers such as FDR for formally verifying properties of policies, and for comparing access control policies with each other (policy subsumption and equivalence). In addition, the authors show how limited workflows can be mapped to CSP, too. The workflow is sequential in nature and in that sense their approach is more expressive than first-order logic approaches.

In [16], the authors propose expressing XACML policies using Multi-Terminal Binary Decision Diagrams (MTBDDs). MTBDDs [17] are a more general version of Binary Decision Diagrams, that map bit vectors over a set of variables to a finite set of results. In [16], variables in the decision diagram are used to represent attribute/value pairs (such as role=Student, action=View, etc.) and the policy results (Deny, Permit, Indeterminate) are mapped to diagram terminals. The paper presents Margrave, a tool for analyzing XACML policies. Margrave provides verification and comprehensive change-impact analysis support based on the semantic differences between the MTBDDs representing the policies.

In [11] the authors extend the work by Fisler et al. [16] by adding more expressiveness to the language being analyzed and supporting additional analysis services. Their tool, called EXAM (comprehensive framework for analysis of access control policies), in addition to supporting core XACML defined in [16] also supports datatype domains. One of the components in the framework is a policy similarity analyzer [39] which is used to filter out policies with low similarity score.

Finally, in [32] a description logic-based approach to analyzing XACML is presented. The authors provide a formalization of XACML that explores the space between propositional logic analysis tools (such as Margrave [16]) and full First-Order logic XACML analysis tools (like Alloy [25]). As a basis for the

XACML formalization they use description logics (DL), which are a family of languages that are decidable subsets of First-Order logic and are the basis for the Web Ontology Language (OWL [13]). Because of the correspondence of policy analysis services to DL reasoning services (e.g., policy inclusion can be reduced to concept subsumption, whereas change impact analysis and verification can be reduced to concept satisfiability), the framework can easily provide a variety of policy analysis services and leverage the availability of off-the-shelf DL reasoners optimized for these services. In addition to the analysis services, their framework provides other benefits: a) the web nature of OWL (it uses URIs for naming and allows for links between ontologies) is suitable for representing an access control language for web resources and b) it supports ontology-based descriptions for subjects,roles and resources used in the policy.

References

1. Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.
2. Dakshi Agrawal, James Giles, Kang-Won Lee, and Jorge Lobo. Policy ratification. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, pages 223–232, Washington, DC, USA, 2005. IEEE Computer Society.
3. Lujo Bauer, Scott Garriss, and Michael K. Reiter. Distributed proving in access-control systems. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 81–95, Washington, DC, USA, 2005. IEEE Computer Society.
4. Moritz Becker, Cedric Fournet, and Andrew Gordon. Design and semantics of a decentralized authorization language. *Computer Security Foundations Symposium, 2007. CSF '07. 20th IEEE*, pages 3–15, 6-8 July 2007.
5. Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, 2001.
6. Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1):71–127, 2003.
7. Jery Bryans. Reasoning about xacml policies using csp. In *SWS '05: Proceedings of the 2005 workshop on Secure web services*, pages 28–35, New York, NY, USA, 2005. ACM Press.
8. C. Ellison and B. Frantz and B. Lampson and R. Rivest and B. Thomas and T. Ylonen . RFC 2693 – SPKI Certificate Theory, 1999.
9. Jan Chomicki, Jorge Lobo, and Shamin Naqvi. A logic programming approach to conflict resolution in policy management. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 121–132, San Francisco, 2000. Morgan Kaufmann.
10. Ernesto Damiani, Sabrina De Capitani di Vimercati, Cristiano Fugazza, and Pierangela Samarati. Extending policy languages to the semantic web. In *ICWE*, pages 330–343, 2004.
11. Dan Lin and Prathima Rao and Elisa Bertino and Jorge Lobo and Ninghui Li. EXAM - a Comprehensive Environment for the Analysis of Access Control Policies, 2007.

12. David E. Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and MULTICS Interpretation. Technical Report MTR-2997, The MITRE Corporation, 1976.
13. Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Web Ontology Language (OWL) Reference Version 1.0. W3C Working Draft 12 November 2002 http://www.w3.org/TR/2002/WD-owl-ref-20021112/.
14. John DeTreville. Binder, a logic-based security language. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 105, Washington, DC, USA, 2002. IEEE Computer Society.
15. Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In *3rd International Joint Conference on Automated Reasoning (IJCAR)*, 2006.
16. Kathi Fisler, Shriram Krishnamurthi, Leo A. Meyerovich, and Michael Carl Tschantz. Verification and change-impact analysis of access-control policies. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 196–205, 2005.
17. M. Fujita, P. C. McGeer, and J. C.-Y. Yang. Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation. *Form. Methods Syst. Des.*, 10(2-3):149–169, 1997.
18. Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, Kent Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *European Semantic Web Symposium*, May 2004.
19. Dimitar P. Guelev, Mark Ryan, and Pierre-Yves Schobbens. Model-checking access control policies. In *ISC*, pages 219–230, 2004.
20. Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. In *In Proceedings of the Computer Security Foundations Workshop (CSFW'03)*, Los Alamitos, CA, USA, 2003. IEEE Computer Society.
21. Joseph Y. Halpern and Vicky Weissman. A formal foundation for xrml. In *CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations*, page 251, Washington, DC, USA, 2004. IEEE Computer Society.
22. Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.
23. C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.
24. Graham Hughes and Tevfik Bultan. Automated verification of access control policies (technical report). Technical Report 2004-22, Department of Computer Science, University of California, Santa Barbara, September 2004.
25. Daniel Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, 2002.
26. Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *Database Systems*, 26(2):214–260, 2001.
27. Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Eliza Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 474–485, New York, NY, USA, 1997. ACM Press.
28. Sushil Jajodia and Duminda Wijesekera. A flexible authorization framework for e-commerce. In *ICDCIT*, pages 336–345, 2004.

29. Trevor Jim. Sd3: A trust management system with certified evaluation. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 106, Washington, DC, USA, 2001. IEEE Computer Society.
30. L. et al Kagal. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, June 2003.
31. Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner. Using semantic web technologies for policy management on the web. In *21st National Conference on Artificial Intelligence (AAAI)*, 2006.
32. Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 677–686, New York, NY, USA, 2007. ACM.
33. Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler. Representing Web Service Policies in OWL-DL. In *Proc. of the Int. Semantic Web Conference (ISWC)*, 2005.
34. Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.
35. Ninghui Li and John Mitchel. Understanding spki/sdsi using first-order logic, 2003.
36. Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.
37. Ninghui Li and Mahesh V. Tripunitara. Security analysis in role-based access control. *ACM Transactions on Information Systems Security*, 9(4):391–420, 2006.
38. Ninghui Li, William H. Winsborough, and John C. Mitchell. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *IEEE Symposium on Security and Privacy*, May 2003.
39. Dan Lin, Prathima Rao, Elisa Bertino, and Jorge Lobo. An approach to evaluate policy similarity. In *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 1–10, New York, NY, USA, 2007. ACM.
40. Fabio Massacci. Reasoning about security: A logic and a decision method for role-based access control. In *First International Joint Conference on Qualitative and Quantitative Practical Reasoning ECSQARU-FAPR*, pages 421–435, 1997.
41. Moritz Becker and Peter Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, page 159, Washington, DC, USA, 2004. IEEE Computer Society.
42. Riccardo Pucella and Vicky Weissman. A Formal Foundation for ODRL. In *In Proceedings of the Workshop on Issues in the Theory of Security (WITS-04)*, 2006.
43. Ronald L. Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.
44. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
45. Andreas Schaad and Jonathan D. Moffett. A lightweight approach to specification and analysis of role-based access control extensions. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 13–22, New York, NY, USA, 2002. ACM Press.
46. Michael Carl Tschantz and Shriram Krishnamurthi. Towards reasonability properties for access-control policy languages. In *SACMAT '06: Proceedings of the*

eleventh ACM symposium on Access control models and technologies, pages 160–169, New York, NY, USA, 2006. ACM Press.

47. A. Uszokand and J. Bradshaw. Kaos policies for web services. In *W3C Workshop on Constraints and Capabilities for Web Servies*, October 2004.
48. Vladimir Kolovski and Bijan Parsia. WS-Policy and Beyond: Application of OWL Defaults to Web Service Policies. In *In Proceedings of the 2nd International Semantic Web Policy Workshop (SWPW'06)*, 2006.
49. Daniel J. Weitzner, Jim Hendler, Tim Berners-Lee, and Dan Connolly. Creating a policy-aware web: Discretionary, rule-based access for the world wide web.
50. Marianne Winslett, Charles C. Zhang, and Piero A. Bonatti. Peeraccess: a logic for distributed authorization. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 168–179, New York, NY, USA, 2005. ACM Press.
51. Thomas Y. C. Woo and Simon S. Lam. Authorizations in distributed systems: A new approach. *Journal of Computer Security*, 2(2-3):107–136, 1993.
52. Nan Zhang, Mark D. Ryan, and Dimitar Guelev. Evaluating access control policies through model checking. In *Eighth Information Security Conference (ISC05)*, 2005.
53. Chen Zhao, NuerMaimaiti Heilili, Shengping Liu, and Zuoquan Lin. Representation and reasoning on rbac: A description logic approach. In *ICTAC*, pages 381–393, 2005.