Semantic Web Policies - A Discussion of Requirements and Research Issues

* P.A. Bonatti[1], C. Duma[2], N. Fuchs[3], W. Nejdl[4], D. Olmedilla[4], J. Peer[5], and N. Shahmehri[2]

[1] Università di Napoli Federico II, Napoli, Italy
`bonatti@na.infn.it`
[2] Linköpings universitet, Linköpings, Sweden
`{cladu,nahsh}@ida.liu.se`
[3] University of Zurich, Zurich, Switzerland
`fuchs@ifi.unizh.ch`
[4] L3S Research Center and University of Hanover, Hanover, Germany
`{olmedilla,nejdl}@l3s.de`
[5] St. Gallen University, St. Gallen, Switzerland
`joachim.peer@unisg.ch`

Abstract. Policies are pervasive in web applications. They play crucial roles in enhancing security, privacy and usability of distributed services. There has been extensive research in the area, including the Semantic Web community, but several aspects still exist that prevent policy frameworks from widespread adoption and real world application. This paper discusses important requirements and open research issues in this context, focusing on policies in general and their integration into trust management frameworks, as well as on approaches to increase system cooperation, usability and user-awareness of policy issues.

Keywords: Integrated heterogeneous policies, Cooperative policy enforcement, Lightweight trust, Trust management, Natural language interfaces, Explanation mechanisms.

1 Introduction

Policies are pervasive in web applications. They play crucial roles in enhancing security, privacy and usability of distributed services, and indeed may determine the success (or failure) of a web service. However, users will not be able to benefit from these protection mechanisms unless they understand and are able to personalize policies applied in such contexts. For web services this includes policies for access control, privacy and business rules, among others.

In this paper, we summarize research performed over the past years on semantic policies and especially aim to analyse those aspects that did not receive so much attention so far. We will focus our discussion on the following strategic goals and lines of research:

* In alphabetical order

- Adoption of a *broad notion of policy*, encompassing not only access control policies, but also privacy policies, business rules, quality of service, and others. We believe that all these different kinds of policies should eventually be integrated into a single framework.
- *Strong and lightweight evidence*: Policies make decisions based on properties of the peers interacting with the system. These properties may be strongly certified by cryptographic techniques, or may be reliable to some intermediate degree with lightweight evidence gathering and validation. A flexible policy framework should try to merge these two forms of evidence to meet the efficiency and usability requirements of web applications.
- These desiderata imply that trust negotiation, reputation models, business rules, and action specification languages have to be integrated into a single framework at least to some extent. It is crucial to find the right tradeoff between generality and efficiency. *So far, no framework has tried to merge all aspects into a coherent system.*
- *Automated trust negotiation* is one of the main ingredients that can be used to make heterogeneous peers effectively interoperate. This approach relies on and actively contributes to advances in the area of *trust management.*
- *Lightweight knowledge representation and reasoning* does not only refer to computational complexity; it should also reduce the effort to specialize general frameworks to specific application domains; and the corresponding tools should be easy to learn and use for common users, with no particular training in computers or logic. We regard these properties as crucial for the success of a semantic web framework.
- The last issue cannot be tackled simply by adopting a rule language. Solutions like *controlled natural language syntax for policy rules*, to be translated by a parser into the internal logical format, will definitively ease the adoption of any policy language.
- *Cooperative policy enforcement*: A secure cooperative system should (almost) never say *no*. Web applications need to help new users in obtaining the services that the application provides, so potential customers should not be discouraged. Whenever prerequisites for accessing a service are not met, web applications should explain what is missing and help the user in obtaining the required permissions.
- As part of cooperative enforcement, advanced *explanation mechanisms* are necessary to help users in understanding policy decisions and obtaining the permission to access a desired service.

In the remainder of this paper we describe the current state of the art on these issues, expand on them and point out several interesting research directions related to them. Section 2 discusses the different types of policies which must be considered in order to address real world scenarios. The need for strong and lightweight evidence on the information that policies require is discussed in Section 3. Section 4 then highlights the importance of trust management as part of a policy framework, describing in detail negotiations and provisional actions. Section 5 describes how cooperative systems which explain their decisions to users as well as policy specification in natural language increase user awareness and understanding. We finish with conclusions and perspectives in Section 6

2 A broad notion of policy

Policies are pervasive in all web-related contexts. Access control policies are needed to protect any system open to the internet. Privacy policies are needed to assist users while they are browsing the web and interacting with web services. Business rules specify which conditions apply to each customer of a web service. Other policies specify constraints related to Quality of Service (QoS). In E-government applications, visas and other documents are released according to specific eligibility policies. This list is not exhaustive and is limited only by the class of applications that can be deployed in the world wide web.

Most of these policies make their decisions based on similar pieces of information [3] – essentially, properties of the peers involved in the transaction. For example, age, nationality, customer profile, identity, and reputation may all be considered both in access control decisions, and in determining which discounts are applicable (as well as other eligibility criteria). It is appealing to integrate these kinds of policies into a coherent framework, so that (i) a common infrastructure can be used to support interoperability and decision making, and (ii) the policies themselves can be harmonized and synchronized.

In the general view depicted above, policies may also establish that some events must be logged (audit policies), that user profiles must be updated, and that when a transaction fails, the user should be told how to obtain missing permissions. In other words, policies may specify *actions* whose execution may be interleaved with the decision process. Such policies are called *provisional policies*. In this context, *policies act both as decision support systems and as declarative behavior specifications.* An effective user-friendly approach to policy specification could give common users (with no training in computer science or logic) better control on the behavior of their own system (see the discussion in Section 5).

Of course, the extent to which this goal can be achieved depends on the policy's ability to *interoperate* with legacy software and data – or more generally, with the rest of the system. Then a policy specification language should support suitable primitives for interacting with external packages and data in a flexible way.

The main challenges raised by these issues are then the following:

– Harmonizing security and privacy policies with business rules, provisional policies, and other kinds of policy is difficult because their standard formalizations are based on different derivation strategies, and even different reasoning mechanisms (cf. Section 4). Deduction, abduction, and event-condition-action rule semantics need to be integrated into a coherent framework, trying to minimize subtleties and technical intricacies (otherwise the framework would not be accessible to common users).

– Interactions between a rule-based theory and "external" software and data have been extensively investigated in the framework of logic-based mediation and logic-based agent programming [18, 17]. However, there are novel issues related to implementing high-level policy rules with low-level mechanisms such as firewalls, web server and DBMS security mechanisms, and operating system features, that are often faster and more difficult to bypass than rule interpreters [14]. A convincing realization of this approach might boost the application of the rich and flexible languages developed by the security community.

3 Strong and lightweight evidence

Currently two major approaches for managing trust exist: policy-based and reputation-based trust management. The two approaches have been developed within the context of different environments and target different requirements. On the one hand, policy-based trust relies on "strong security" mechanisms such as signed certificates and trusted certification authorities (CAs) in order to regulate access of users to services. Moreover, access decisions are usually based on mechanisms with well defined semantics (e.g., logic programming) providing strong verification and analysis support. The result of such a policy-based trust management approach usually consists of a binary decision according to which the requester is trusted or not, and thus the service (or resource) is allowed or denied. On the other hand, reputation-based trust relies on a "soft computational" approach to the problem of trust. In this case, trust is typically computed from local experiences together with the feedback given by other entities in the network. For instance, eBay buyers and sellers rate each other after each transaction. The ratings pertaining to a certain seller (or buyer) are aggregated by eBay's reputation system into a number reflecting seller (or buyer) trustworthiness as judged by the eBay community. The reputation-based approach has been favored for environments such as Peer-to-Peer or Semantic Web, where the existence of certifying authorities can not always be assumed but where a large pool of individual user ratings is often available.

Another approach – very common in today's applications – is based on forcing users to commit to contracts or copyrights by having users click an "accept" button on a pop-up window. This is perhaps the lightest approach to trust, that can be generalized by having users utter *declarations* (on their e-mail address, on their preferences, etc.) e.g. by filling an HTML form.

Real life scenarios often require to make decisions based on a combination of these approaches. Transaction policies must handle expenses of all magnitudes, from micropayments (e.g. a few cents for a song downloaded to your iPod) to credit card payments of a thousand euros (e.g. for a plane ticket) or even more. The cost of the traded goods or services contributes to determine the risk associated to the transaction and hence the trust measure required.

Strong evidence is generally harder to gather and verify than lightweight evidence. Sometimes, a "soft" reputation measure or a declaration in the sense outlined above is all one can obtain in a given scenario. We believe that the success of a trust management framework will be determined by the ability of *balancing trust levels and risk levels* for each particular task supported by the application, adding the following to the list of interesting research directions:

- How should different forms of trust be integrated? Some hints on modelling context aware trust, recommendation and risk with rules is given in [16] and a first proposal for a full integration in a policy framework can be found in [6]. However, new reputation models are being introduced, and there is a large number of open research issues in the reputation area (e.g., vulnerability to coalitions). Today, it is not clear which of the current approaches will be successful and how the open problems will be solved. Any proposal should therefore aim at maximal modularity in the integration of numerical and logical trust.
- How many different forms of evidence can be conceived? In principle, properties of (and statements about) an individual can be extracted from any – possibly unstructured – web resource. Supporting such a variety of information in policy decisions is a typical semantic web issue – and an intriguing one. However, such general policies are not even vaguely as close to become real as the policies based on more "traditional" forms of evidence (see the discussion in the next section).

4 Trust management

During the past few years, some of the most innovative ideas on security policies arose in the area of *automated trust negotiation* [2, 7, 5, 9, 22–25, 1]. That branch of research considers peers that are able to automatically negotiate credentials according to their own declarative, rule-based policies. Rules specify for each resource or credential request which properties should be satisfied by the subjects and objects involved. At each negotiation step, the next credential request is formulated essentially by *reasoning* with the policy, e.g. by inferring implications or computing abductions.

Since about five years frameworks exist where credential requests are formulated by exchanging *sets of rules* [7, 5]. Requests are formulated *intensionally* in order to express compactly and simultaneously all the possible ways in which a resource can be accessed — shortening negotiations and improving privacy protection because peers can choose the best option from the point of view of sensitivity. It is not appealing to request "*an ID and a credit card*" by enumerating all possible pairs of ID credentials and credit card credentials; it is much better to *define* what IDs and credit cards are and send the definition itself. Another peer may use it to check whether some subset of its own credentials fulfills the request. This boils down to gathering the relevant concept definitions in the policy (so-called *abbreviation rules*) and sending them to the other peer that reasons with those rules locally.

In [7, 5] *peers communicate by sharing their ontologies.* Interestingly, typical policies require peers to have a common a priori understanding only of the predicate representing credentials and arithmetic predicates, as any other predicate can be understood by sharing its definition. The only nontrivial knowledge to be shared is the X.509 standard credential format. In this framework, interoperability based on ontology sharing is already at reach! This is one of the aspects that make policies and automated trust negotiation a most attractive application for semantic web ideas.

Another interesting proposal of [5] is the notion of *declaration*, that has already been discussed in Section 3. This was the first step towards a more flexible and lightweight approach to policy enforcement, aiming at a better tradeoff between protection efforts and risks. According to [15], this framework was one of the most complete trust negotiation systems. The major limitation was the lack of distributed negotiations and credential discovery, which are now supported as specified in [7].

Negotiations In response to a resource request, a web server may ask for credentials proving that the client can access the resource. However, the credentials themselves can be sensitive resources. So the two peers are in a completely symmetrical situation: the client, in turn, asks the server for credentials (e.g. proving that it participates in the Better Business Bureau program) before sending off the required credentials. Each peer decides how to react to incoming requests according to a local policy, which is typically a set of rules written in some logic programming dialect. As we pointed out, requests are formulated by selecting some rules from the policies. This basic schema has been refined along the years taking several factors into account [2, 7, 5, 9, 22–25, 1].

First, policy rules may possibly inspect a *local state* (such as a legacy database) that typically is not accessible by other peers. In that case, in order to make rules intelligible to the recepient, they are partially evaluated with respect to the current state.

Second, *policies themselves are sensitive resources*, therefore not all relevant rules are shown immediately to the peer. They are first filtered according to policy release rules; the same schema may be applied to policy release rules themselves for an arbitrary but finite number of levels. As a consequence, some negotiations that might succeed, in fact fail just because the peers do not tell each other what they want. The study of methodologies and properties that guarantee negotiation success is an interesting open research issue.

Moreover, *credentials are not necessarily on the peer's host.* It may be necessary to locate them on the network [11]. As part of the automated support to *cooperative enforcement*, peers may give each other hints on where a credential can be found [26].

There are further complications related to actions (cf. Section 4). In order to tune the negotiation strategy to handle these aspects optimally, we can rely on a *metapolicy language* [7] that specifies which predicates are sensitive, which are associated to actions, which peer is responsible for each action, and where credentials can be searched for, guiding negotiation in a declarative fashion and making it more cooperative and interoperable. Moreover, the metapolicy language can be used to instantiate the framework in different application domains and link predicates to the ontologies where they are defined.

Provisional policies Policies may state that certain requests or decisions have to be logged, or that the system itself should search for certain credentials. In other words, policy languages should be able to specify *actions*. Event-condition-action (ECA) rules constitute one possible approach. Another approach consists in labelling some predicates as *provisional*, and associating them to actions that (if successful) make the predicate true [7]. We may also specify that an action should be executed by some other peer; this results in a request.

A cooperative peer tries to execute actions under its responsibility whenever this helps in making negotiations succeed. For example, provisional predicates may be used to encode business rules. The next rule[6] enables discounts on low_selling articles in a specific session:

$$\begin{aligned}&\texttt{allow}(\mathit{Srv}) \leftarrow \ldots, \texttt{session}(\mathit{ID}),\\ &\quad \texttt{in}(X, \texttt{sql:query}('\texttt{select} * \texttt{from low\_selling}')),\\ &\quad \texttt{enabled}(\texttt{discount}(X), \mathit{ID})\,.\end{aligned}$$

Intuitively, if $\texttt{enabled}(\texttt{discount}(X), \mathit{ID})$ is not yet true but the other conditions are verified, then the negotiator may execute the action associated to `enabled` and the rule becomes applicable (if $\texttt{enabled}(\texttt{discount}(X), \mathit{ID})$ is already true, no action is executed). The (application dependent) action can be defined and associated to `enabled` through the metapolicy language. With the metalanguage one can also specify when an action is to be executed.

Some actions would be more naturally expressed as ECA rules. However, it is not obvious how the natural bottom-up evaluation schema of ECA rules should be integrated with the top-down evaluation adopted by the current core policy language. The latter fits more naturally the abductive nature of negotiation steps. So integration of ECA rules is still an interesting open research issue.

Stateful vs. stateless negotiations Negotiations as described above are in general stateful, because (i) they may refer to a local state – including legacy software and data – and (ii) the sequence of requests and counter requests may become more efficient if credentials and declarations are not submitted again and again, but kept in a local negotiation state. However, negotiations are not *necessarily* stateful because

- the server may refuse to answer counter-requests, or – alternatively – the credentials and declarations disclosed during the transaction may be included in every message and need not be cached locally;
- the policy does not necessarily refer to external packages.

[6] formulated in Protune's language

Stateless protocols are just special cases of the frameworks introduced so far. Whether a stateless protocol is really more efficient depends on the application. Moreover, efficiency at all costs might imply less cooperative systems.

Are stateful protocols related to scalability issues? We do not think so. The web started as a stateless protocol, but soon a number of techniques were implemented to simulate stateful protocols and transactions in quite a few real world applications and systems, capable of answering a huge number of requests per time unit. We observe that if the support for stateful negotiations had been cast into http, probably many of the intrinsic vulnerabilities of simulated solutions (like cookies) might have been avoided.

New Issues Existing approaches to trust management and trust negotiation already tackle the need for flexible, knowledge-based interoperability, and take into account the main idiosyncrasies of the web – because automated trust negotiation frameworks have been designed with exactly that scenario in mind. Today, to make a real contribution (even in the context of a policy-aware web), we should further perform research on the open issues of trust management, including at least the following topics:

- Negotiation success: how can we guarantee that negotiations succeed despite all the difficulties that may interfere: rules not disclosed because of lack of trust; credentials not found because their repository is unknown. What kind of properties of the policy protection policy and of the *hints* (see Section 4) guarantee a successful termination when the policy “theoretically” permits access to a resource?
- Optimal negotiations: which strategies optimize information disclosure during negotiation? Can reasonable preconditions prevent unnecessary information disclosure?
- In the presence of multiple ways of fulfilling a request, how should the client choose a response? We need both a language for expressing preferences, and efficient algorithms for solving the corresponding optimization problem. While this negotiation step is more or less explicitly assumed by most approaches on trust negotiation, there is no concrete proposal so far.

Additionally, integration of abductive semantics and ECA semantics is an open issue, as we have pointed out in a previous section.

5 Cooperative policy enforcement

Cooperative enforcement involves both machine-to-machine and human-machine aspects. The former is handled by negotiation mechanisms: published policies, provisional actions, hints, and other metalevel information (see Section 4) can be interpreted by the client to identify what information is needed to access a resource, and how to obtain that information.

Let us discuss the human-machine interaction aspect in more detail: One of the most important causes of the enormous number of computer security violations on the Internet is the users' lack of technical expertise. Users are typically not aware of the security policies applied by their system, neither of course about how those policies can be changed and how they might be improved by tailoring them to specific needs. As a consequence, most users ignore their computer's vulnerabilities and the corresponding countermeasures, so the system's protection facilities cannot be effectively exploited.

It is well known that the default, generic policies that come with system installations – often biased toward functionality rather than protection – are significantly less secure than a policy specialized to a specific context, but very few users know how to tune or replace the default policy. Moreover, users frequently do not understand what the policy really checks, and hence are unaware of the risks involved in many common operations.

Similar problems affect privacy protection. In trust negotiation, credential release policies are meant to achieve a satisfactory tradeoff between privacy and functionality – many interesting services cannot be obtained without releasing some information about the user. However, we cannot expect such techniques to be effective unless users are able to understand and possibly personalize the privacy policy enforced by their system.

A better understanding of a web service's policy makes it also easier for a first-time user to interact with the service. If denied access results simply in a "*no*" answer, the user has no clue on how he or she can possibly acquire the permission to get the desired service (e.g., by completing a registration procedure, by supplying more credentials or by filling in some form). This is why we advocate *cooperative policy enforcement*, where negative responses are enriched with suggestions and other explanations whenever such information does not violate confidentiality (sometimes, part of the policy itself is sensitive).

For these reasons, *greater user awareness and control on policies* is one of our main objectives, making policies easier to understand and formulate to the common user in the following ways:

– Adopt a *rule-based policy specification language*, because these languages are flexible and at the same time structurally similar to the way in which policies are expressed by nontechnical users.
– Make the policy specification language more friendly by e.g. developing a *controlled natural language* front-end to translate natural language text into executable rules (see next section).
– Develop *advanced explanation mechanisms* [4, 12, 13] to help the user understand what policies prescribe and control.

Inference Web (IW) [12, 13] is a toolkit that aims at providing useful explanations for the behavior of (Semantic-) Web based systems. In particular, [12] propose support for knowledge provenance information using metadata (e.g., Dublin Core information) about the distributed information systems involved in a particular reasoning task. [12] also deals with the issue of representing heterogeneous reasoning approaches, domain description languages and proof representations; the latter issue is addressed by using PML, the OWL-based Proof Markup Language [8].

Specifically applied to policies, [4] contains a requirements analysis for explanations in the context of automated trust negotiation and defines explanation mechanisms for *why, why-not, how-to*, and *what-if* queries. Several novel aspects are described:

– Adoption of a *tabled explanation structure* as opposed to more traditional approaches based on single derivations or proof trees. The tabled approach makes it possible to describe infinite failures, which is essential for *why not* queries.
– Explanations show simultaneously different possible proof attempts and allow users to see both local and global proof details at the same time. This combination of local and global (intra-proof and inter-proof) information facilitates navigation across the explanation structures.
– Introduction of suitable heuristics for focussing explanations by removing irrelevant parts of the proof attempts. A second level of explanations can recover missing details, if desired.
– Heuristics are *generic*, i.e. domain independent, they require no manual configuration.
– The combination of tabling techniques and heuristics yields a novel method for explaining failure.

Explanation mechanisms should be *lightweight* and *scalable* in the sense that (i) they do not require any major effort when the general framework is instantiated in a specific application domain, and (ii) most of the computational effort can be delegated to the clients. Queries are answered using the same policy specifications used for negotiation. Query answering is conceived for the following categories of users:

– Users who try to understand how to obtain access permissions;
– Users who monitor and verify their own privacy policy;
– Policy managers who verify and monitor their policies.

Currently, advanced queries comprise *why/why not, how-to*, and *what-if* queries. Why/why not queries can be used by security managers to understand why some specific request has been accepted or rejected, which may be useful for debugging purposes. Why-not queries may help a user to understand what needs to be done in order to obtain the required permissions, a process that in general may include a combination of automated and manual actions. Such features are absolutely essential to enforce security requirements without discouraging users that try to connect to a web service for the first time. How-to queries have a similar role, and differ from why-not queries mainly because the former do not assume a previous query as a context, while the latter do.

What-if queries are hypothetical queries that allow to predict the behavior of a policy before credentials are actually searched for and before a request is actually submitted. What-if queries are good both for validation purposes and for helping users in obtaining permissions.

Among the technical challenges related to explanations, we mention:

- Find the right tradeoff between explanation quality and the effort for instantiating the framework in new application domains. Second generation explanation systems [19–21] prescribe a sequence of expensive steps, including the creation of an independent domain knowledge base expressly for communicating with the user. This would be a serious obstacle to the applicability of the framework.

Natural language policies Policies should be written by and understandable to users, to let them control behavior of their system. Otherwise the risk that users keep on adopting generic hence ineffective built-in policies, and remain unaware of which controls are actually made by the system is extremely high – and this significantly reduces the benefits of a flexible policy framework.

Most users have no specific training in programming nor in formal logics. Fortunately, they spontaneously tend to formulate policies as rules; still, logical languages may be intimidating. For this reason, the design of front ends based on graphical formalisms as well as *natural language interfaces* are crucial to the adoption of formal policy languages. We want policy rules to be formulated like: "*Academic users can download the files in folder historical_data whenever their creation date precedes 1942*".

Clearly, the inherent ambiguity of natural language is incompatible with the precision needed by security and privacy specifications. Solutions to that can be the adoption of a *controlled* fragment of English (e.g., the ATTEMPTO system[7]) where a few simple rules determine a unique meaning for each sentence. This approach can be complemented with a suitable interface that clarifies what the machine understands.

6 Conclusions and perspectives

Policies are really knowledge bases: a single body of declarative rules used in many possible ways, for negotiations, query answering, and other forms of system behavior control. As far as trust negotiation is concerned, we further argue that transparent interoperation based on ontology sharing can become "everyday technology" in a short time, and trust negotiation especially will become a success story for semantic web ideas and techniques.

In addition to stateless negotiation (see [10]), we need stateful negotiation as well [5]. Even the Web, which started as a stateless protocol, now implements a number of techniques to simulate stateful protocols and transactions, especially in applications for accessing data other than web pages.

Cooperative policy enforcement and trust management gives common users better understanding and control on the policies that govern their systems and

[7] `http://www.ifi.unizh.ch/attempto/`

the services they interact with. The closer we get to this objective, the higher the impact of our techniques and ideas will be.

Policies will have to handle decisions under a wide range of risk levels, performance requirements, and traffic patterns. It is good to know that the rule-based techniques that different research communities are currently converging to are powerful enough to effectively address such a wide spectrum of scenarios. This is the level of flexibility needed by the Semantic Web.

Acknowledgment

This research has been partially funded by the European Commission and by the Swiss State Secretariat for Education and Research within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`).

References

1. M. Y. Becker and P. Sewell. Cassandra: distributed access control policies with tunable expressiveness. In *5th IEEE International Workshop on Policies for Distributed Systems and Networks*, Yorktown Heights, June 2004.
2. M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in the PolicyMaker Trust Management System. In *Financial Cryptography*, British West Indies, February 1998.
3. P. A. Bonatti, N. Shahmehri, C. Duma, D. Olmedilla, W. Nejdl, M. Baldoni, C. Baroglio, A. Martelli, V. Patti, P. Coraggio, G. Antoniou, J. Peer, and N. E. Fuchs. Rule-based policy specification: State of the art and future work. Technical report, Working Group I2, EU NoE REWERSE, aug 2004. `http://rewerse.net/deliverables/i2-d1.pdf`.
4. P.A. Bonatti, D. Olmedilla, and J. Peer. Advanced policy queries. Technical Report I2-D4, Working Group I2, EU NoE REWERSE, Aug 2005. http://www.rewerse.net.
5. P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002. Short version in the Proc. of the Conference on Computer and Communications Security (CCS'00), Athens, 2000.
6. Piero A. Bonatti, Claudiu Duma, Daniel Olmedilla, and Nahid Shahmehri. An integration of reputation-based and policy-based trust management. In *Semantic Web Policy Workshop in conjunction with 4th International Semantic Web Conference*, Galway, Ireland, nov 2005.
7. Piero A. Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, pages 14–23, Stockholm, Sweden, jun 2005. IEEE Computer Society.
8. Paulo P. da Silva, Deborah L. McGuinness, and Richard Fikes. A proof markup language for semantic web services. Technical Report KSL Tech Report KSL-04-01, January, 2004.
9. Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic*

Web Symposium (ESWS 2004), volume 3053 of *Lecture Notes in Computer Science*, pages 342–356, Heraklion, Crete, Greece, may 2004. Springer.

10. Vladimir Kolovski, Yarden Katz, James Hendler, Daniel Weitzner, and Tim Berners-Lee. Towards a policy-aware web. In *Semantic Web Policy Workshop in conjunction with 4th International Semantic Web Conference*, Galway, Ireland, nov 2005.
11. N. Li, W. Winsborough, and J.C. Mitchell. Distributed Credential Chain Discovery in Trust Management (Extended Abstract). In *ACM Conference on Computer and Communications Security*, Philadelphia, Pennsylvania, November 2001.
12. Deborah L. McGuinness and Paulo Pinheiro da Silva. Explaining answers from the semantic web: The inference web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.
13. Deborah L. McGuinness and Paulo Pinheiro da Silva. Trusting answers from web applications. In *New Directions in Question Answering*, pages 275–286, 2004.
14. Arnon Rosenthal and Marianne Winslett. Security of shared data in large systems: State of the art and research directions. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 962–964. ACM, 2004.
15. K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobsen, H. Mills, and L. Yu. Requirements for Policy Languages for Trust Negotiation. In *3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, June 2002.
16. Steffen Staab, Bharat K. Bhargava, Leszek Lilien, Arnon Rosenthal, Marianne Winslett, Morris Sloman, Tharam S. Dillon, Elizabeth Chang, Farookh Khadeer Hussain, Wolfgang Nejdl, Daniel Olmedilla, and Vipul Kashyap. The pudding of trust. *IEEE Intelligent Systems*, 19(5):74–88, 2004.
17. V. S. Subrahmanian, Piero A. Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, and Robert Ross. *Heterogenous Active Agents.* MIT Press, 2000.
18. V.S. Subrahmanian, S. Adali, A. Brink, R. Emery, J.J. Lu, A. Rajput, T.J. Rogers, R. Ross, and C. Ward. Hermes: Heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/publications/ abstracts/hermes. html.
19. William Swartout, Cecile Paris, and Johanna Moore. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert: Intelligent Systems and Their Applications*, 6(3):58–64, 1991.
20. Michael C. Tanner and Anne M. Keuneke. Explanations in knowledge systems: The roles of the task structure and domain functional models. *IEEE Expert: Intelligent Systems and Their Applications*, 6(3):50–57, 1991.
21. M. R. Wick. Second generation expert system explanation. In J.-M. David, J.-P. Krivine, and R. Simmons, editors, *Second Generation Expert Systems*, pages 614–640. Springer Verlag, 1993.
22. W. Winsborough, K. Seamons, and V. Jones. Negotiating Disclosure of Sensitive Credentials. In *Second Conference on Security in Communication Networks*, Amalfi, Italy, September 1999.
23. W. Winsborough, K. Seamons, and V. Jones. Automated Trust Negotiation. In *DARPA Information Survivability Conference and Exposition*, Hilton Head Island, SC, January 2000.
24. Marianne Winslett, Ting Yu, Kent E. Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.

25. Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.
26. C. Zhang, P.A. Bonatti, and M. Winslett. Peeraccess: A logic for distributed authorization. In *12th ACM Conference on Computer and Communication Security (CCS 2005)*, Alexandria, VA, USA. ACM.