Agents, Trust, and Information Access on the Semantic Web

Tim Finin and Anupam Joshi
Department of Compute Science and Electrical Engineering
University of Maryland Baltimore County
{finin, joshi}@cs.umbc.edu

Information systems are becoming increasingly important and ubiquitous in our lives -- for organizations, professionals, individuals and families. The process began with large centralized computers and greatly accelerated with the development of inexpensive personal computers and user friendly software systems. The slope increased with the rapid deployment of the Internet and corporate databases in the 80's and 90's and exploded with the unfolding of the Web in the past ten years. While it's hard to make predictions, many expect the trend to quicken with continued advances in mobile computing, ad hoc wireless networking, microelectronics, and nanotechnology. Imagine a world with billions of people and agents who interact daily with billions of computational devices, each of which is consuming and producing information and communicating with scores of other devices on a constant and largely autonomous basis. This evolution provides many new challenges to our ability to design and deliver information systems.

This paper describes our work in transforming the web from a "universal database" model to an active ecology of agents that produce, consume, and act on information (semi) autonomously. In particular, we focus on an immediate and critical problem which must be addressed to make progress -- our ability to maximize security, privacy, and trusted behavior in an environment which is fundamentally dynamic, open and devoid of many of the clues human societies have relied on for trust assessment. The approach is to explore the role that "semantic web" languages can play in addressing these problems.

I. Introduction

Today's web is designed for direct use by people, who must view, understand, select and navigate its information. The semantic web of the future is intended for software agents as well as humans. This requires that the information on the web be encoded in ways that facilitate understanding and manipulation by computer programs. This encoding is achieved through the specification and utilization of ontologies – formal representations of a domain. The shift enabled by the use of machine understandable ontologies, will be from a web which only humans can understand, to one which **both** humans and machines can understand. Until this shift occurs, endeavors that require finding data spread out across the web or dynamically drawing inferences based on this data will continue to be hampered by their reliance on ad-hoc, task specific frameworks. Consider the following scenario from the not too distant future.

John is on his way to work when he receives a call from his sister-in-law saying that his wife is in the emergency room at St. Josephs. John tells his personal agent to get directions to St. Josephs and to postpone all his meetings. John's agent goes to the web to locate information about St. Josephs and finds two hospitals named St. Josephs in the area. The agent queries the agents of both hospitals to find out which hospital John's wife was admitted to. After this, the agent uses the Mapquest agent to get directions to the hospital and informs the car's navigation agent. John also has a meeting in 20 minutes with Paul from the Sales department. John's agent contacts Paul's scheduling agent and informs it that John cannot attend due to a family emergency. Paul's agent displays this message to Paul on his PDA and asks Paul if it should reschedule the meeting.

Paul is busy and does not see the message. Paul's agent and John's agent make a tentative

appointment for the next day after verifying that it does not cause any conflicts, and enter this into John's and Paul's calendars. They inform John and Paul about the meeting.

As we see from this scenario, in the near future there will be an active ecology of agents on the semantic web that will collaborate with each other and access information sources on the web to fulfill users' needs. The focus of much of the existing semantic web work [Cost02, horrocks02] has been on the creation of languages for knowledge representation such as DAML+OIL and OWL [OWL02]. Other efforts have been to create tools to help annotate web pages with semantic markup [kalyanpur02, kogut01], or efficient techniques to index semantic information[Sheth02].

However, a key need for the vision of semantic web to succeed will be the ability for handling security, privacy and trust in the process of information retrieval. In the above example, John's agent needs to query the hospitals agent to retrieve information about John's wife. Not all agents should be able to do this, but its important that John be able to find out where his wife is admitted. How can the hospital's agent decide who should be given access to this sensitive information? John's agent should also be able to communicate with John's car and upload directions. Similarly Paul's agent and John's agent should be able to communicate and schedule meetings together. In short, with the advent of the semantic web, as information is used more and more by agents and not human users, security becomes increasingly important, and it is crucial for security mechanisms to be embedded into the fabric of the semantic web and be as automated as possible.

The obvious solution is to extend security mechanisms applicable to distributed systems (e.g. Kerberos, PKI, SPKI, etc.) for the semantic web. However, we believe that this will be difficult given the completely decentralized nature of the web, the extremely large number of agents and users, and their heterogeneity. In addition, the set of users/agents that access an information source or interact with a given agent cannot be enumerated *a priori*. Along with this is the problem of semantic meaning of the security information; it is not feasible to expect all entities to use the same terminology to represent security protocols and information. A security framework for the web needs to be flexible, semantically rich, impervious to common network problems like network partitioning, and simple enough to automate.

II. Overview

The eBiquity group at UMBC (http://research.ebiquity.org/) has been working on several issues related to the use of semantic web technologies in the data and information management. We provide a brief overview of these next. The subsequent sections detail our work on trust and belief as mechanisms for security.

1. Dynamic Composition of Data and Services. Much of the data and information available on the semantic web will be in the form of services -- self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Examples of existing services include stock quotes, hotel reservations, airline ticket sales, etc. In the near future, sensors and embedded devices will also provide data streams as services. Our approach is to describe these services using the DAML-S ontology for higher level task specific issues, coupled with the DReggie[SRDS01] ontology for lower level system specific issues.

Sometimes, services need to be sequenced or composed in order to provide true value. We wish to enable researchers to say "I have data consisting of a million records of [description of data], where can I go for a three dimensional visualization of the correlations in my data." Such a service may not be available. However, perhaps a service is available which takes as input multivariate data correlations (in some specified format), and gives as output visualizations, while another will compute the correlations given the data set. We are developing a system capable of identifying and composing the data and computation services required to provide a composite service [PWC02,Chen01].

2. Indexing the Hidden Web & Semantic Markup. Search engines— Google, Infoseek, etc. —work by constantly crawling the Web and building huge indexes, with entries for every word encountered. However, they ignore the tags. While in HTML tags are simply formatting information, ignoring tags in the semantic web means loosing the DAML information on the page! One simple approach we are trying is to see if the DAML markup can be made to appear as "text" to the search engine. The information will then be indexed by the search engine. The user will then be allowed to query on not just arbitrary free text, but also on structured attributes of the information expressed in DAML. The query engine uses the same approach to convert the structured part of the query into text, and then gives the "combined" free text to the search engine. We have implemented such a system with text retrieval systems built by UMBC and JHU, and the preliminary results are very encouraging[Shah02].

A related issue is that much information on the web is "hidden" behind forms. For example, http://animaldiversity.ummz.umich.edu/ allows the user to search a vast collection of articles on members of the animal kingdom. But it does not *link* to those articles. The user is required to enter a search term.

A Web-spider, not knowing how to interact with such sites, cannot penetrate any deeper than the page holding the form. Now imagine that http://animaldiversity.ummz.umich.edu/ was marked up using DAML, which said "I am http://animaldiversity.ummz.umich.edu/. I am an interface to a vast collection of articles about animals. If you input an animal species name into the box, I will return a list of articles containing the following information about the species: geographic range, preferred habitat, etc."

We are developing a data model to enable such communication. A search spider that understands this data model will be able to query the site directly, and act accordingly. A DAML aware spider can come to such a page and do one of two things:

- It can say to itself: "I know some species names. I'll input them (being careful not to overwhelm the site), and index the results (and keep on spidering from the result pages).
- The search engine can record the fact that the site returns articles about the species whose name is entered via a form

III. Distributed Trust and Belief.

Not all of the data on the internet are reliable. When a human being is searching the web, it is relatively easy for her to determine whether or not to trust a particular web page. For example, she may only trust scientific claims made by professors at universities. Or, if she is in a controversial field, her criteria may be even more restrictive. That is, there may be only a handful of researchers whose judgments she trusts on matters relating to her field. Managing trust in this distributed environment is thus an important issue. If a software agent is going to respond to her queries by assimilating data distributed throughout the web, her trust preferences will need to be transmitted to the agent. For example, "On AIDS related research questions, believe any site trusted by Anthony Fauci." or "On biodiversity related research questions, trust anyone trusted by E. O. Wilson." The flip side to the question "Which information sources should my agent believe?" is "Which agents should I allow to access my information source?" For example, allow everyone who is a faculty (or represents them) at UMBC to access the ACM digital library content.

We believe that in order to secure the Semantic Web two main components are required: a semantic policy language for defining security requirements and a distributed trust management approach. As the Semantic Web is composed of web resources (e.g., web pages, web services, agents, and human users), security should be uniformly applied to, and be applicable to, all. It is important for web entities to be able to express their security and belief information in a clear and concise manner so that its meaning is unambiguous. Towards this end we suggest the use of a policy language based on a semantic language to markup security information. This policy language will provide constructs for users, agents and web resources to clearly define their security requirements.

In traditional security systems, security includes some form of authentication followed by access control. However these techniques do not scale well, and generally require a central server for authorization. They suffer from overhead in communication and also make several implicit trust assumptions. Access control in these distributed scenarios is also a problem because storing this information in a central location is not always feasible or desirable. Hence it no longer makes sense to divide authorization into authentication and access control. We assume an open environment in which web entities must interact with agents and web resources with which they are not familiar. In particular, an entity will receive requests and assertions from other entities and must decide how to act on the requests and assess the credibility of the assertions. In a closed environment, web entities have well known and familiar transaction partners whose rights and credibility are known. The problem thus reduces to authentication -- the reliable identification of entities' true identity. In an open environment, however, entities must transact business even when knowing the true identities is uninformative. Decisions about whom to believe and whom to serve must be based on an entity's properties. These properties are established by proving them from an entity's credentials, delegation assertions, and the appropriate security policy.

Blaze, et. al.[blaze96] introduced the term distributed trust management in 1996 and defined it as creating policies, assigning credentials to users, and checking if the credentials of the requester conform to the policy before granting it access. We believe that distributed trust management overcomes several deficiencies of current security mechanisms by providing access control to a large number of resources from a large number of users that may be foreign, providing security without necessarily authenticating users completely, providing flexibility in specifying security requirements and giving every entity a certain amount of autonomy in making their own security decisions. In particular, the trust notions introduced by Blaze can be extended by the notion of delegation and locally specified policies to create security for the semantic web.

Trust is explained in terms of a relationship between a trustor, who trusts a certain entity, and a trustee, the trusted entity. Based on her trust in the trustee, a trustor can decide whether the trustee should be allowed to access her resources and what rights should be granted, or if it should believe the information/services that the trustee claims to provide. Therefore, trust plays an important role in deciding both the access rights as well as provenance of information. Trust management involves using the trust information, including recommendations from other trustees, to reason about these two issues. In dynamic systems, every entity enforces its individual policies so a trust management component cannot use global policies or trust relationships to evaluate requests for authorization. This causes trust pertaining to each entity to be handled specifically in terms of its own policies, allowing the entire process to be decentralized.

In its security aspects, our approach is similar to role based access control - in that a user's access rights are computed from his/her properties. However, we propose to use not just role hierarchies but any properties and constraints expressed in a semantic language including elements of both description logics and declarative rules. This would allow us to describe the security policies including delegation policies for each resource. Consider the earlier example of an agent believing that sites trusted by Fauci on AIDS issues should be assumed reliable. What if some such a site makes the assertion that sites under the NIH domain should be trusted as well. Should the agent now add to its beliefs this trust delegation? Or consider the case of UMBC faculty being allowed to access the ACM Digital Library. Should a UMBC faculty be allowed to delegate this right to a student, and should the ACM DL honor this delegation? The delegation is of course controlled – one may delegate all or a part of one's rights. Delegation also allows rights to be assigned dynamically to a person, who has no role within some organization, without creating a new short term role specific

to this person. For instance, a UMBC faculty could delegate the right to access the ACM DL to a visitor for a few hours, without the visitor having any predefined role (faculty, student, staff etc.) in the UMBC organization. Similarly, rights can be revoked from a user without changing his/her role, making this approach more flexible and maintainable than traditional Role based methods.

A delegation is a granting of a right from one entity (delegator) to another entity (delegatee). A delegation consists of various information; delegator, right, constraints on delegatee, constraints on execution, constraints on re-delegation and time period. Only agents with the ability to delegate can make valid delegations. An agent has the ability to make any delegation, but whether it is honored depends on various factors, including the security policy, the agent's rights, and the rights of the agents ahead it in the delegation chain. Only authorized delegations actually change access rights of other agents, the others are voided. The right to delegate is defined implicitly and explicitly. Implicitly, an agent can delegate rights to any service it offers and explicitly, an agent that has been given the right to delegate by another agent, who has the right, can perform valid delegations, as long as the delegation fulfills the constraints of the previous delegation. This forms a chain of constraints; the agent at the end of the chain must satisfy all the constraints associated with the delegations in the chain. Our delegation mechanism makes sure of this, before allowing an agent to access a service. Revocation in a delegation chain causes all the delegations after the revocation to be invalidated.

To prevent insecure/illegal delegations, our framework supports the addition of constraints to delegations. By using constraints on the delegatee, the delegator can specify who it can delegate. Constraints on re-delegation allow the delegator to decide whether the right can be re-delegated and to whom it can be re-delegated. For example, Myra could give her agent the ability to use her credit card number and the right to give this ability to all members of her immediate family. We have developed rules that capture this information and enforce security by checking these constraints at the redelegation and run time.

IV. An Ontology for Policies

As we argued earlier, each entity must be able to describe its security requirements and trust information in an intelligible manner, so that the meaning is distinct. To provide this functionality, we need a language for describing security policies and trust information based on a semantic language like RDF-S or OWL[OWL02]. Using the axiomatic semantics of the underlying language and the policy language, the entities will be able to interpret the information correctly.

The policy language will have some domain independent ontologies but will also require specific domain ontologies. The former includes concepts for permissions, obligations, actions, speech acts, operators etc. that an entity uses to define security policies. The latter is a set of ontologies, shared by the agents in the system, which defines domain classes (person, file, runJavaScript, readHTML) and properties associated with the classes (age, num-pages, email). We have defined a preliminary language that uses deontic concepts of rights, obligations, prohibitions and dispensations that we refer to as policy objects. Associated with each policy object is a set of conditions (credentials) that an entity has to satisfy in order to possess that policy object. For example, if an agent only allows agents from its own company to access its services, associated with the *right to access services* policy object will be the set of conditions that belong to every agent in its company. The language includes certain specifications for credentials, for example, *logging in required*, or *digital certificate required*, but most of the requirements will be domain dependent and will be provided by the application developer/owner of resource/agent. An engine will reason over these policy objects and the credentials of the requester to decide the requester's access rights. We allow conditions to be domain dependent, along with action specifications, e.g., accessing a service, inserting into a database etc.

The language allows speech acts to be modeled and provides a way of specifying conversation policies. As agents are going to be very common on the Semantic Web, we believe that speech acts will be used extensively for communication. We will study the effect of these speech acts on the trust information, e.g. an utterance involving a delegation causes changes in the trust relationships if it is valid.

Another aspect of policies is conflict resolution, i.e. resolving conflicting policies. This is something we have not yet addressed seriously, but will build into meta-policies. These decide how policies are interpreted. There are several ways of handling conflicts. For instance, we are investigating allowing policy makers to explicitly give precedence to certain policies over others. Similarly, where resources are in a hierarchy (e.g. university, college, department, research group), the policies of entities higher in the hierarchy could supersede conflicting policies in lower entities. Our ontology at present restricts itself to explicitly identifying trusted entities. We are working to extend it to allow the characteristics of trusted sources to be specified. For example, enable preferences like "Trust university professors when they make claims in their field." Ideally, we would also like to allow these preferences to be abstracted from a user's on-line behavior. Another related issue is the binary nature of trust at present – an entity is either trusted or not. We'd like to extend our ontology, and the corresponding reasoning mechanisms, to deal with degrees of trust as well.

V. Conclusions

In this paper we present our ongoing research on enabling agent based information access from the semantic web in a "secure" manner. We present a broad overview of our work, and also details on the distributed trust based approach to security and privacy we have adopted.

References

[Cost02] R. Scott Cost, Tim Finin, Anupam Joshi, Yun Peng, Charles Nicholas, Filip Perich, Harry Chen, Lalana Kagal, Youyong Zou, and Sovrin Tolia, ***ITtalks: A Case Study in the Semantic Web and DAML+OIL*** , IEEE Intelligent Systems , January/February, 2002.

[horrocks02] Ian Horrocks et al., ***DAML+OIL Language Specifications***, http://www.daml.org/

[OWL02] W3C, ***OWL Web Ontology Language 1.0 Reference***, http://www.w3.org/TR/2002/WD-owl-ref-20020729/, July 2002

[kalyanpur02] Aditya Kalyanpur and James Hendler, ***RDF Web Scraper v1.1***, http://www.ece.umd.edu/~adityak/running.html, 2002.

[kogut01] Paul Kogut and William Holmes, ***AeroDAML: Applying Information Extraction to Generate DAML annotations from Web Pages,*** First International Conference on Knowledge Capture, Workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, 2001.

[Sheth02] S. Thacker, A. Sheth and S. Patel, "***Complex Relationships for the Semantic Web,***" Creating the Semantic Web, D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster (eds.), MIT Press, 2002.

[SRDS01] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and Anupam Joshi, ***DReggie: Semantic Service Discovery for M-Commerce Applications***, Workshop on Reliable and Secure Applications in Mobile Environment, 20th Symposium on Reliable Distributed Systems, October 28-31, 2001.

[Chen01] Chen, H., Joshi, A., Finin, T., "***Dynamic Service Discovery for Mobile Computing: Intelligent Agents meet Jini in the Aether***", *Baltzer Science Journal on Cluster Computing* , Special Issue on Advances in Distributed and Mobile Systems and Communications, 4 :4, pp 343-354, October 2001.

[PWC02] Dipanjan Chakraborty, Filip Perich, Anupam Joshi, Timothy Finin, Yelena Yesha, "***A Reactive Service Composition Architecture for Pervasive Computing Environments***", In 7th Personal Wireless Communications Conference (PWC 2002). Singapore. October. 2002.

[Shah02] Urvi Shah, Tim Finin, Anupam Joshi,R. Scott Cost, and James Mayfield, ***Information Retrieval on the Semantic Web,*** accepted for publication in ACM CIKM 2002, McLean, VA.

[Blaze96] M. Blaze, J. Feigenbaum, J. Lacy, ***Decentralized Trust Management***, Proc. IEEE Conf. Privacy and Security, 1996.