Of Web Trust and Policies: A suggested framework to enhance Internet Security

Steven Caruana
Department of Computer Science
Univeristy of Malta
scar009@um.edu.mt

Dr. Matthew Montebello
Department of Computer Science
Univeristy of Malta
mmont@cs.um.edu.mt

ABSTRACT

We tend to trust people, software or anything else around us through experience or through a recommendation from a trusted source. Web voyeurs have similarly envisaged the notion of software entities roaming over the World-Wide Web capable of trusting other similar entities. Ideally, Web agents would be able to distinguish and differentiate between sites, services and resources over the Internet that are reliable and worth the confidence accredited to them. In this paper we present several trust and policy frameworks built within the evolving agent mark-up languages in an attempt to encapsulate the Web in a new dimension of trust. Furthermore, we propose a novel mechanism that exploits existent policies, that govern servers and provide them with credentials responsible for their authentication, by extending the existent web site structure.

Keywords

Agents, Semantic Web, Trust, Security, Policies

1. INTRODUCTION

Trust is one word which is important to the greater part of humanity for whatever the scope, reason or context one applies it. Most often business, industry, academics and researchers place information and the divulgation of information as the centre point of all of their activities. This means that people for whom the evolution of means of communication and dissemination of information plays a central role for their operations and accomplishments, trust gains more significance. With the onset of more complex and demanding e-services in various fields, the concerns of security has become a reason for which many a brow will furrow. In a recent talk during a World-Wide Web Consortium [16] meeting in London, Sir Tim Berners Lee pointed out that People learn to trust through experience and through recommendation, and argues that a Web of trust would be not only safe for people to use and access but also for software which needs reliability and dependability when servicing user requests. The notion of having trustworthy web services such as active networks, mobile agents, etc. which thrive upon entire distributed networks of information and computation resources, is becoming more challenging as requests are becoming more complex and less trivial. The rest of the paper is organised as follows: In Section 2 we review how trust over the Web is currently being achieved through the use of a number of policy frameworks. In Section 3 we propose a model for building trust over the Semantic Web through the use of entities and adapting policies to govern those entities. The model described in Section 3 aims to empower users with creating their own policy frameworks, merging them with distributed authenticated systems which are essential components for achieving trust and ensuring security over the web. The last section of the paper describes the future implementation of the application and the extent of research needed in the field of trust and security over the World Wide Web.

2. TRUST OVER THE SEMANTIC WEB

The current web has no notion of entities or relationships between entities. Users browse the Internet finding information in the same document format, style and layout as they are used to handling. On the other hand, software agents whose role is to handle information and service requests by the users, have no means of parsing information from the traditional layout. For these agents to be able to crawl the web and service user requests, there needs to be defined a service-oriented architectural structure to represent data on the internet and the relations that different clusters of information display. The Semantic Web is a possible solution for this problem because it deals with the representation and aggregation of relationships between resources on the web.

One of the more important factors that tend to dissuade the community from the adoption of Semantic Web Applications is the notion of trust. At this present stage in the development of the Semantic Web we have produced languages (DAML + OIL, OWL[18] etc.) which are capable of representing information on the internet. Despite the richness of these languages we are still not able to describe relations like "I want to trust only papers written by professors who are still registered with a university and these papers must have been published less than five years ago", or "Do not give information to users unless they can clear their identity as members of the university I am registered with".

2.1 Introducing Trust

Research projects use the term trust to signify different meanings. The following is one such definition:

> "Trust is a method of dealing with uncertainty; when dealing with independent agents, institutions or providers of resources (including knowledge), one trusts them if one accepts their characterisation of what they will do. Trust can be a moral notion (X trusts Y to act in Xs interests), or not (X trusts Y to perform some task T)." [12]

If one were to apply the above statement to the Semantic Web it would mean that a user (human or agent) has the capacity to trust other users, but that not every user can be trusted. In this paper we are defining trust as being either of two issues. The first is the issue of authenticating a person as being whom they claim to be. Trust can also be defined as being a measure of the degree of trust that a user attributes to another user in the context of a particular domain.

Most of these problems are not new to the internet. Projects such as the Microsoft .Net Passport[11] and OpenID[13] have provided solutions for possible distributed authentication infrastructures. On the other hand projects such as Foaf[6] have provided us with a means of adding our credentials in a machine readable format across the web.

2.2 Trust and Policy Frameworks

When the Semantic Web was first introduced in [1] one of the ideas which was put forward was that of having a web of correlated information from which software agents could acquire knowledge which they would eventually use to help the user for every day life service requests. In turn research projects have been making progress in trying to find solutions for various issues that this vision put forward. As a result of this, trust frameworks were developed by research groups and are now being used by developers researching the field of semantics in web applications.

The following section will give a brief description of few such projects.

2.2.1 Ponder

Ponder[3] is a distributed policy management system that was developed a few years back. This system was designed to allow users across the web to specify information (policies) about resources which would otherwise be impossible to define and enforce.

These specifications were recorded using what the authors referred to as policies. Ponder supports a number of policy specifications, including *Authorisation* policies, *Obligation* policies, *Refrain* policies, *Delegation* policies, *Composite* policies and *Meta-policies*. Each of these policies can have *constraints* applied to them. These policies are then applied to specific domains for which Ponder provides the authentication and trust reasoning engine.

2.2.2 P3P

P3P[17] is a project organised by the W3C[16] which is trying to provide a specification for an extension to the current web. This initiative aims at enabling web sites to store information about privacy policies which should be applied to them. This information is meant to be analysed and processed by agents that can use this information in the background on the user's behalf, while he/she is browsing other web pages. Even though this information is meant to be used by agent software, this standard also caters for human users by presenting information in a reader-friendly way. Having agents process privacy preferences on behalf of the user will conveniently take the load off the user, relieving him/her of having to acknowledge every privacy policy for each individual page.

Numerious implementations of this standard have already been presented and can be found at the W3C web site[16].

2.2.3 KAoS

KAoS[2] is an open agent system that offers possible solutions to some of the problems that adversely affect agent architectures. In [15] the author writes about how KAoS policies can now be defined in OWL and gives a good description of how KAoS proposes to use a policy ontology termed the KAoS Policy Ontology (KPO). The KPO is then used as a base ontology to define ontologies that can represent statements like:

> It is forbidden for employees of a company X or employees of a company Y to apply for this package.

The system that employs this policy can then use it to determine who is eligible to apply for the package in question and who is not. At runtime the system will then interact with users and once each user registers with the services, the site will be able to formulate who are employees of companies X and Y and determine what they can apply for.

KAoS can be said to be more agent oriented than others because it was designed with agents in mind rather than web users. It aims to provide agents (KAoS agents) with a roaming space intended for authenticating users when required and providing other services which the user might benefit from.

2.2.4 Rei

Rei[7] is another attempt to create a policy language. It provides an ontology which is used to define the policies in its engine. In the Me-Centric project[7], the author proposes a system whereby the use of policies defines a global perspective which is made up of domains and sub domains that can overlap each other. Similar to other projects, this project aims at making this web accessible not only to users but also to software agents.

The first problem that Rei tries to tackle is the fact that users who are not very technical find it hard to understand most policy languages. Therefore Rei is based on First-Order Logic (FOL), which is not only easy for users to read but is also simple to translate to RDF or DAML + OIL. This

means that FOL language can be translated into a semantic language representation.

This system also provides a set of ontologies which are not domain specific so that whoever is writing his/her own policies can use these ready made concepts to create varied operations and functions such as setting of permissions, obligations, speech acts etc. It might be the case that a single domain might require more specific objects to be defined (such as person, readFile, deleteUser etc.) and properties associated with them (name, age, comany, email), which is why Rei also supports the extension and definitions of its policy ontologies by the user.

The Me-Centric policy server stores all the policies that determine how entities are to be treated, while its domain server contains the information pertaining to various domains. The policy server can retrieve the policies of different domains and then use the domain server to map the domain specific names to the policies. When a user requests a speech act, the policy engine will determine the logic of this speech act and then will add the information it gathers to its knowledge base. These speech acts can be categorised in four different policies, *delegate*, *revoke*, *cancel* and *request*.

2.2.5 Rein

Rein[8] is a project that tries to extend on Rei by adding support for N3 logic reasoning. The target audience of Rein is the internet in general and it proposes a few twists to the structures seen so far. All the policy languages reviewed so far have defined their own structures (policy languages) in which policies should be defined (policy files, KAoS ontology, Rei Onotlogy etc.).

This methodology proposes the use of Rei ontologies to define the policies, but unlike other projects, it does not place strict rules to enforce which language is used to define these policies. It promotes the use of different policy languages and refers to policies written in these languages as Meta-Policies. It also takes advantage of various features of the Semantic Web, such as allowing an entity to be categorised by another entity, according to its definition. In such an eventuality the Rein engine will go to the desired URIs and collect the necessary information to complete the definition of the entity.

[9] gives the reader a good idea of how Rein can be used to deploy a domain with varied implementations of Rein and a variety of policies residing on each node. It also gives the reader a good insight as to how Rein ontology is defined and how the Rein engine works. [10] also discusses issues related to the implementation of Rein and adds a few more examples which provide a useful insight as to how a network using Rein policies should be constructed.

3. FROM WEB OF TRUST TO A SEMANTIC WEB OF TRUST

When one browses the web, he/she will encounter a number of pages which encapsulate certain unique features. Some sites propose mechanisms for the sharing and annotation of information (e.g. Flickr[5], Riya[14] etc.) or even community voting which can determine the trust level of a resource or user (e.g. ebay[4] etc.). In this paper we are proposing the construction of a mechanism which extends the current website structure by providing a direct feed to the user's machine providing a diagrammatic representation of the policies that govern the currently viewed website. It is also a means of informing the client's machine about how to provide its user's credentials to the servers that are responsible for authentication. We are also proposing the extension of a typical web server allowing it to make use of a policy language such as Rein to define policies that will govern the specified domain and to provide a means of allowing the server to use distributed authentication.

Below is a diagram of how such a network would be setup and of how the user is expected to interact with the system.

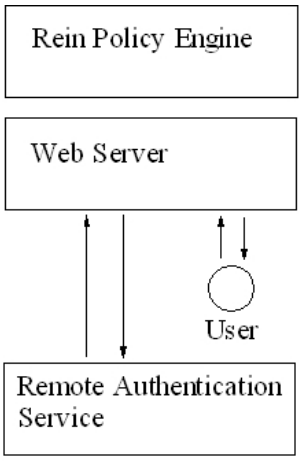

Figure 1: Network Setup

In the diagram above one can see all the major entities that partake in this system. At the back-end there is a server that will await requests from the client. Once the server receives a request it can make use of remote servers to authenticate the client and send an adequate response in return.

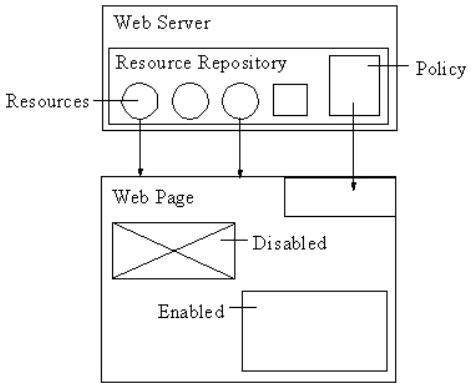

Figure 2: Document transfer

In Figure 2 the browser will display only items for which the user has credentials. Items for which no credentials are given will be automatically blocked. It also shows how the user can send his credentials to provide authentication for the blocked items.

This setup is quite often adopted by those who develop distributed authentication systems. What distinguishes this setup and makes it unique is the way the client and server communication is handled. When the client first requests information from a page the server will return and display the requested data. Along with this, the user will also receive a 'feed' that will inform the browser that the user was not authorised to view certain components because he/she did not fulfil the necessary policy requirements. The user will then be prompted by the browser to allow the use of credentials which would have already been inputted into the system in order to fulfil these requirements. Once this structure is in place, the system will know that this user can be trusted and that his/her credentials can be reused at a later stage if required.

The above proposition could offer a solution in terms of the server which would otherwise have to decide whether the user can be trusted or not. However this still does not solve the problem of how the user could gain the trust of other users viewing pages submitted by him/her. As a solution to this problem, we are proposing that the servers store the credentials within the policy feed attributed to this page, and that agents on the client-side will be able to process this information and make use of rules that the user would have defined beforehand to determine whether a site could be trustworthy or not.

Below is a typical example of how such a system could be used:

A user might log into his/her account on the University of Malta web server. His/her agent submits his credentials to the web server and this in turn keeps track of them. The user then decides to upload a paper about his main area of research. This paper is then forwarded to a board who is assigned to approve it. Once the document is approved, the user's agent returns to the site and updates the credentials which are related to the paper and the policies governing it, thus including the fact that the document was accepted and published.

To extend this example let us consider the added circumstance during which, three other uses are roaming the net looking for papers in this same research area. Once the page is accessed by all three search agents, the privacy field embedded in the page will interoperate with these agents. User agent 'A' has a policy of not showing unpublished papers, whilst 'B' and 'C' lack this policy. When 'A' identifies that this paper has not yet been published it refuses to view the paper and informs the user that an unpublished paper was refused. User agents 'B' and 'C' try to access the paper but the paper requires that to view the paper, the users *must* be on the auditing board. User agent 'B' had been informed a priori that its owner would be reviewing this paper, and sends over the necessary credentials. On the other hand user agent 'C' lacks this credential and sends a message to the user's browser to ask the user to submit the necessary credentials if s/he intends to view this page. User 'C' then inputs an encrypted link to his board credentials and updates his public profile which his agent can then use to re submit the credentials required to view this site.

3.1 Of entities and policies

In this paper we are proposing that web pages and the Internet should have a means of publishing or of enforcing a set of policies. In the example provided in section 3 we can notice that there are various entities and each have their own policies.

The following is a table listing the policies applied in the example found in Section 3

Table 1: Table of entity policies

Entity	Policy
Web Page	Allow all
Paper	If user can be authenticated as being on the submission board
	(After paper approved) Allow all
User Agent A	Do not download publications which have not been approved
User Agent B	Allow all
User Agent C	Allow all

The following is a table listing the credentials found in the example found in Section 3

Table 2: Table of entity credentials

Entity	Credentials
Web Page	Credentials that point to author
Paper	Credentials that point to author
User Agent A	No Credentials in this example
User Agent B	Credentials to prove he/she is a board member
User Agent C	No Credentials
	(After inputting credentials) Credentials to prove he/she is a board member

3.2 Empowering users

An important factor which is to be considered is that it will be hard for such a system to be adopted unless the users are able to migrate their current security structures and perform maintenance on them more efficiently. To assist users in overcoming this problem we are proposing that a browser extension for web authors be developed and that this be used for both testing and updating of the policy files. This extension should provide two different modes of use which are *browse mode* and *edit mode*.

When the user is viewing the page in browse mode the browser should be searching for policy feeds that web pages might be broadcasting and offer the necessary credentials to them accordingly, whilst still offering the user a chance to

replace these credentials with more specialised ones. If the browser cannot find the necessary credentials in the user's profile, then a means of asking the user to input his authentication details should be provided.

In edit mode the browser should visualise the policy feed that the user is constructing, in a logical and intuitive structure. Making it easy to construct and view policy files is of great importance because it will be only the minority of users who are Semantic Web aware and capable of constructing complex policy strucutres manually. A visual aid, such as a graphical user interface, would help bridge this problem by providing the less technical users with an adequate mechanism enabling them avoid dealing with the back-end system whilst still being able to declare their own policies.

Using the structure discussed we would be empowering the less technical users (such as blog users or wiki eidtors) to declare their own policies and help them to implement a Semantic Web Policy Structure to limit access to the resources they publish on the internet. As for the users who are meant to access these web resources, they can use the simple interface provided by their browser to fulfill the requirements that the authors would have set and gain access to the services being broadcasted.

3.3 Distributed Authentication

Distributed authentication research projects have been undergoing development for a number of years. As a by-product of these research groups, products like OpenID[13], Microsoft .Net Passport[11] were developed.

Both of the projects mentioned above aim at offering the user a single sign-on structure. The .Net Passport framework provides a number of Microsoft-owned servers on which each user has an account. Using this infrastructure Microsoft are pushing forward the idea that a user should have the facility for logging into a website (e.g. logging into hotmail) and not needing to log into any other page including ones in separate domains. OpenID tries to push forward a similar concept by offering an infrastructure where the user need only have an account on one domain and should use this account to authenticate himself accross the web.

These systems aim at providing solutions to the problems of authentication and trust. However not one of them provides a solution which could solve both of these problems. The .Net Passport does not offer a real solution for trust problems because even though each user should be assigned a single user login on the internet, it promotes no means of understanding the user's profile. For example if a user has a .Net Passport he/she can login using that passport and roam about on the internet using this credential, but if the user is publishing a paper to a website and this user has already submitted a number of publications before, the .Net Passport has no means of relating these publications.

OpenID on the other hand offers a structure that is not only a single login system but provides an infrastructure that can be used to identify a user as being a member of a certain domain for example, if one decides to submit a blog post on a blog hosted at www.myBlog.com, but his/her current blogging account is found at www.hisBlog.com, using OpenID the user can be allowed to log into hisBlog.com account and post a comment on someone else's www.myBlog.com account. As the system logs the user and authenticates his credentials it is also giving access to the site's security gateway. OpenID does not let you carry forth information about your account to use as credentials for trust algorithms. For example to make a distinction between two classes in a domain (e.g. a profressor of science and a student of science) using OpenID, the only workaround to represent this differnce is for the two classes to be assigned different domains to log into (e.g. student login student@student.cs.um.edu.mt, professor login professor@professor.cs.um.edu.mt)). Although this work around does provide a rudimentary means of attributing trust, it can become very hard to manage, and will only allow the representation of very basic differences between the classes of users and more complex relations (such as number of publications, issued or amount of time dedicated to students a week). If the user were to write the policy files manually with no visual tool they would be next to impossible to keep track of. Policies would need to be amended every time a change in a user's credentials occurs.

In this paper we are proposing a structure that can leverage the power of distributed login mechanisms with an added extension which integrates a trust framework into it. This would allow a user not only to authenticate himself/herself as being registered to a specific domain (like OpenID) but also to carry forward his/her credentials which can give him/her credibility over the web. Using Semantic Web technologies credentials can now be represented in a format that can be reasoned upon (such as rdf) and this can then be used as input for a trust engine (such as Rein) giving as a result the trustworthiness of a website or user.

4. FUTURE WORK

The application being described in this paper is currently still being developed. Further extensions to the mechanisms and structures mentioned above are still being refined.

In this paper we have also emphasised the concept that a ubiquitous security framework might not be the best way around providing security mechanisms. However more research is needed to devise a framework which could bridge the transfer of user credentials across domains using a predefined security standard, whilst still allowing the separate domains to make use their own means of authentication mechanisms.

Finally an idea which will need to be looked into further, is that of providing support for Semantic Web Services and Semantic Web Applications. As the web continues to progress, web services and web enabled applications are becoming ever more vital for the web and its users. Extending such a framework to support the use of policy feeds could provide these applications with a subsystem that caters for trust and authentication.

5. REFERENCES

[1] T. Berners-Lee, J. A. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[2] J. Bradshaw, S. Dutfield, P. Benoit, and J. Wooley.

Software Agents, chapter KAoS: Towards industrial strength open agent architecture, pages 375–418. MIT Press, 1997.

[3] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. Technical report, 2000.

[4] ebay. *http://www.ebay.com*.

[5] Flickr. *http://www.flicker.com*.

[6] FOAF. *http://www.foaf-project.org/*.

[7] L. Kagal. Rei: A policy language for the me-centric project. Hp labs technical report, hpl-2002-270, HP Labs, 2002.

[8] L. Kagal and T. Berners-Lee. Where policies meet rules in the semantic web. Technical report, MIT, 2005.

[9] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Self-describing delegation networks for the web. *IEEE Workshop on Policy for Distributed Systems and Networks (IEEE Policy)*, June 5 - 7 2006.

[10] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Using semantic web technologies for policy management on the web. *21st National Conference on Artificial Intelligence (AAAI)*, July 16 - 20 2006.

[11] Microsoft. *http://www.passport.net*.

[12] K. O'Hara, H. Alani, Y. Kalfoglou, , and N. Shadbolt. Trust strategies for the semantic web. In *Proceedings of the Trust, Security and Reputation workshop at the ISWC04, Hiroshima, Japan*, Nov. 2004.

[13] OpenID. *http://openid.net/*.

[14] Riya. *http://www.rija.com*.

[15] A. Uszok, J. M. Bradshaw, M. Johnson, and R. Jeffers. Kaos policy management for semantic web services. *IEEE INTELLIGENTSYSTEMS*, 284(5):32–41, July-August 2004.

[16] W3C. *http://www.w3.org/*.

[17] W3C. *http://www.w3.org/P3P*.

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