Privacy preserving trust authorization framework using XACML

Mbanaso, UM, Cooper, GS, Chadwick, DW and Proctor, S

Privacy Preserving Trust Authorization Framework Using XACML

U.M.Mbanaso, G.S.Cooper
Informatics Research Inst,
University of Salford, UK
U.M.Mbanaso@pgr.salford.ac.uk

D.W.Chadwick
Computer Department
University of Kent UK
D.W.Chadwick@Kent.ac.uk

Seth Proctor
Sun Microsystems Labs
Burlington, MA, USA
Seth.proctor@sun.com

Abstract

Nowadays many organizations share sensitive services through open network systems and this raises the need for an authorization framework that can interoperate even when the parties have no pre-existing relationships. Trust Negotiation is the process used to establish these first relationships, through the transfer of attributes, embedded in digital credentials, between the two parties. However, these attributes may themselves be considered sensitive and so may need protection from disclosure. In some environments, the policies that govern the protected services may also be considered sensitive and their release to arbitrary strangers may leak confidential business information. This paper describes a way to unify the protection of services, sensitive credentials and policies in a synchronized trustworthy manner. We propose a trust authorization framework (TAF) that builds on the capabilities of XACML to support the bilateral exchange of policies and credentials through trust negotiation.

1. Introduction

Authorization ensures that resources can be accessed only by parties who have the right privileges. Thus, the resource gatekeeper requires some level of trust be established before sensitive information can be released. Service requesters are required to submit sufficient authorization credentials before access will be granted. Wherever people are involved in the exchange of digital information, such as personal, potentially sensitive credentials, privacy [15] [2] [7] becomes an issue of some concern, which raises an interesting paradox. To make the services and resources accessible to legitimate users an authorization infrastructure requires the users' attributes. However, the users may not be ready to disclose their attributes to a remote service provider without determining exactly who the provider is and how their personal attributes will be used. One approach for addressing these privacy concerns is to employ a bilateral exchange of policies and credentials between the parties involved in the transaction, so that they can decide what to give and/or get from each other. This process is known as trust negotiation in the literature [8].

Consider the following motivating example. A Secret Service (SS) offers online training both for its agents and friendly secret agent services. The service requires that each participant present a role Attribute Certificate (certificate), a security assertion digitally signed by the participant's security authority, which binds the holder's attributes to the holder. Whilst the policy that governs this service prevents unauthorized access to its resources, it does not protect the fact that SS offers training to friendly organizations, which is itself a sensitive piece of business information.

To prevent arbitrary disclosure of sensitive policies, access to the policies themselves needs to be protected. On the other hand, an agent requester cannot give out her role certificate to any service that poses as the SS web server, and would like some proof that the server can be trusted. To avoid the arbitrary disclosure of sensitive policies [2] and digital credentials, parties require a mechanism to gradually establish a trust relationship. Trust relationships can be established between service providers and requesters through the exchange of information in a well-understood fashion [10]. The information usually contains policies and security assertions, issued by Attribute Authorities (AAs), which describe the properties of the holders. The exchange of this information is done in such a manner that the security assertions are unforgeable and can be verified and validated [18].

Trust negotiation management systems have been proposed by researchers as one effective way to guarantee the confidentiality of authorization information. Trust establishment is a well-researched concept [5] [2] [6]. However, existing efforts in this area have not been standardized and do not fit into any authorization standard such as the eXtensible Access Control Markup Language (XACML) [3], which would provide the benefit of promoting interoperability and reducing the effort needed to integrate with existing applications. This work investigates how

Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)
0-7695-2593-8/06 $20.00 © 2006 IEEE

XACML can fit into trust authorization management systems by exploring existing concepts, and where necessary, extending them to accomplish our goal.

We describe our proposed XACML Trust Authorization Framework (XTAF). XTAF is a loosely coupled architecture with a trust component that protects authorization information (policies and credentials) layered such that it integrates seamlessly into any XACML compliant authorization engine with minimal effort. We expose different ways that the XACML policy language can be used to support bilateral exchange of policies and credentials, and protect unauthorized access to services. We introduce a Trust Authorization Service Handler (TASH) to handle trust and privacy of authorization information. This supports runtime bilateral authorization operations between two or more parties.

The rest of this paper is organized as follows: In section 2, we provide an overview of related work, highlighting some of the challenges. Section 3 gives a brief overview of the XACML authorization framework and the proposed XACML Trust Authorization Framework. In section 4 we illustrate, through a hypothetical example, the usage of our framework. Section 5 concludes the paper with a summary.

2. Related Works

Seamons et el [13] [14] [11] [12] and Bertino et el [6] [8] have done useful works in the area of Trust Negotiation and Management, providing a good theoretical background on the concepts of trust with quite a number of implementation scenarios. Seamons et el have advanced the notion of a trust negotiation protocol and strategy with some practical demonstration of how they can be implemented [10]. In the area of trust policy and language, Bertino et el have proposed a number of ways to encode policies and credentials [6]. However, these works are proprietary and cannot interoperate; thus investigating how the XACML framework can fit into trust management systems becomes important.

Lorch et el [4] presented their first experience using XCAML in distributed systems, including the analysis of the performance of XACML with existing models, and highlighted its limitations. They drew on experience gained in the integration of SAML [22] and XACML in distributed open systems and performance results based on the PRIMA model [19]. PRIMA is specifically designed for access control in grid computing environments: users can assign and/or delegate privileges to each other without involving policy administrators. However, Lorch et el focused mainly on the analysis of XACML's performance and did not address the privacy issues and trustworthiness in distributed environments. Our work is among the first to look into how XACML can be used to build trust relationships in distributed authorization environments. This is a significant direction since XACML is a generic access control model that has continued to address wider access control requirements.

The Shibboleth infrastructure, an attempt to address privacy in an authorization environment, proposed two kinds of policies: Attribute Release Policy (ARP) and Attribute Acceptance Policy (AAP) [16]. Shibboleth is a distributed authentication architecture whose access control is based on users' attributes. Privacy in Shibboleth is primarily focused on using pseudonymity; however this does not completely protect privacy in an environment where the user may give other attributes in order to use the authorized resources. For instance, an institution may give a student a signed assertion, authorization token to access a discount online bookshop. But if the student wants to purchase a book, (s)he needs to provide other personal attributes such as credit card number, physical address for payment and delivery. In such a case, the user cannot determine whether a party can be trusted with sensitive attributes. Lorch et el also used WSPL, a profile of XACML that supports policy intersection to determine whether two policies are mutually amenable. This approach requires some policy on the server side to be released without negotiation, but then provides a very simple means for calculating what the client is willing to share.

PERMIS [11] [18] is a middleware authorization framework, which focuses mainly on the role based access control (RBAC) model. PERMIS has successfully been implemented in a number of application scenarios with interesting results [19][20] [18]. It fully supports role hierarchy and its policy language is user friendly. It has a GUI policy editing tool [21] and Privilege Allocation (PA) subsystems for managing roles and permissions. The PERMIS language is limited in expressions and semantics compared to XACML. The PERMIS framework does not provide direct support for bilateral exchange of policies and credentials to address privacy issues. PERMIS has in its architecture a subsystem that signs, verifies and validates X.509 attribute certificates used to represent authorization credentials.

3. XACML Trust Authorization Framework

One promising mechanism to solve privacy and trust is the eXtensible Access Control Markup Language (XACML), a standard created in OASIS [3]. This standard defines a general-purpose, flexible authorization policy language and a query/response format. The XACML standard uses a generic access control framework based on the IETF/DMTF model that allows an enterprise to specify and deploy an access control policy for a variety of resources. Though XACML is a rich framework, it intentionally does not address how to preserve the privacy of authorization entities.

An architecture is required that can support trust and confidentiality at the same time. Access control techniques can be used to protect access to a party's credentials, but to establish trust requires a gradual and progressive approach in the exchange of a party's credentials. This entails a bilateral process, in which both parties can use access control policies to determine the way their attributes are given to each other, requiring a repeated exchange of policies and credentials as trust is progressively increased. In order to know which credentials to release, a subject must be sent a policy of the resource. If the subject is happy with the policy, it will release further credentials. Rather than taking all the risk of releasing sensitive attributes at once, parties are subject to smaller risk on an incremental basis and are able to withdraw at any point.

3.1 Trust Authorization Architecture

Figure 1 shows the basic building block of XACML Trust Authorization Framework. Core XACML components are described in [3]. We introduce a Trust Authorization Service Handler (TASH), a component added to the core XACML model to address the aspects of privacy and trust in distributed authorization environments. In the normal XACML approach, the Policy Decision Point (PDP) requests attribute values from the ContextHandler. In theory, the ContextHandler can query the Policy Information Point (PIP) for the attributes; in SunXACML, the AFM (Attribute Finder Module) [17] does the job of finding attributes that were not in the initial request context. The proposed service is being implemented as a Trust Negotiation (TN) server integrated via a SunXACML AFM to the core XACML engine, allowing the TASH to work seamlessly with the XACML engine. In TASH, the Negotiation Protocol Module (NPM) handles the trust negotiation protocols and ordering of messages [10] during the building of a trust relationship. The Attribute Validation Engine (AVE) verifies and validates every credential attribute and policy that is received by the system before passing it to the trust decision engine. The Trust Information Handler (TIH) is responsible for the canonical representation of the inputs consumed by the TrustPDP and the outputs from it.

The TrustPDP handles trust access management decisions by comparing local policies with received credentials and received policies with local credentials. The TrustPDP performs trust access management decisions in two ways:

- It checks whether there are any local credentials (and policies) that can be disclosed by comparing the received credentials with the local policy. This is a necessary but not sufficient step for releasing further local credentials (and policies).
- It checks the received policy to see whether there is sufficient benefit to be gained from releasing further local credentials. When the recipient is a human user, he or she can be asked to make a decision. When the recipient is a service being accessed by a user, then there may be no received policy but it may still be beneficial to the service to release further local credentials and policies.

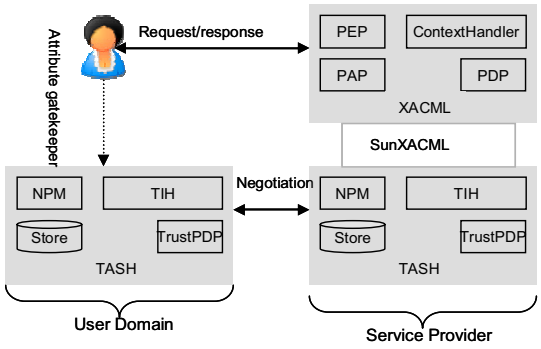

Figure 1. XACML Trust Authorization Architecture

Both parties in the exchange require a TASH in order to engage in a trust building session.

3.2 XACML Trust Policy Set

The XACML language provides several ways to form a negotiation policy set. To enable the gradual building of trust, we arrange access control policies as directed policy graphs or trees [10], enabling the sequence and ordering of disclosure policies to be discovered at runtime. A node at any level is a pre-

condition for evaluating that branch of the graph and for continuing to process the other parts of the graph. Similarly, in XACML, each *Target* at any node of the tree is an intersection of *Targets* in the path that leads to that branch of the tree. This demands that if the *Target* at any level evaluates false, evaluating that branch of the tree becomes needless.

We examine two pragmatic ways in which the XACML policy language can be used to form an effective trust negotiation policy set. One approach is to use the existing *PolicySet* container depicted in figure 2, which can be considered as a tree containing one or more children: *PolicySet* or *Policy*. A *Policy* on the other hand contains one or more child elements: *Rule,* and a *PolicyId* attribute. Thus, in *PolicySet,* each *Policy* can specify a disclosure policy. The sequence and ordering can be determined by using the notion of *PolicyIds*, to specify the order in which the policies are disclosed at runtime. We give a simple example here. Alice wants to access *webserver1* protected by policy *p2*, which specifies that the subject must be a nuclear research student in the computing department of the University of Salford. We assume that *p2* is considered sensitive, so that its disclosure is controlled by another policy *p1*. It can only be disclosed to a subject with a proof of affiliation with the University of Salford. We can implicitly specify a generic requirement in the *PolicySet Target,* then *p1* and *p2* as policies in the *PolicySet,* but with *p2*'s *policyId* as the attribute of the protected Resource in *p1*. Here, *p2* is disclosed only if

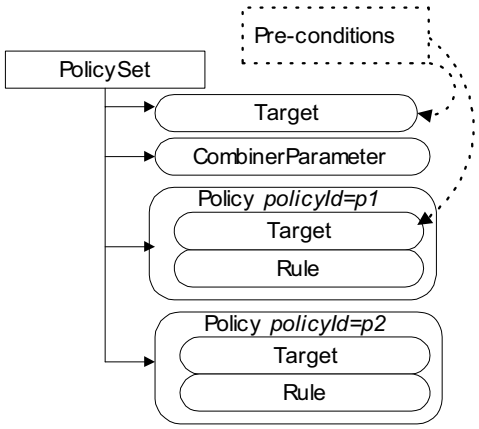

Figure 2. The XACML PolicySet

p1 is evaluated to true. Thus, the idea of a *PolicySet*, in theory can be used to construct effective trust negotiation policy set, whose order can be determined, but requires good crafting skills. However, this introduces a computational overhead, in that processing of the policy requires the evaluation of two pre-conditions: *PolicySet Target* and *Policy Target*, before the *Policy Rule* is evaluated.

A second approach is to make use of the *RuleId* attribute of the *Rule* container, since a *Policy* can contain one or more *Rule* elements, as shown in figure 3. In this case, a *Rule* can be made to point to another *Rule* in order to protect that *Rule* from disclosure to arbitrary strangers using the *RuleId*. The sequence and order of disclosure can be determined simply by finding the relationships between the *Rule* containers. This model is less complex, and has less computational overhead. Again, *Rule* containers can be used to express fine-grained disclosure policies, and additional constraints can be enforced in each rule by using the XACML Conditions and Obligations in a more refined way. We adopt this model as an efficient way to construct a simple, effective trust policy set. Though, in some environments, the first approach can be more effective, especially where policies are defined by a hierarchy of authorities to protect authorization information flow.

At trust session runtime, an effective disclosure trust policy set is constructed from the applicable

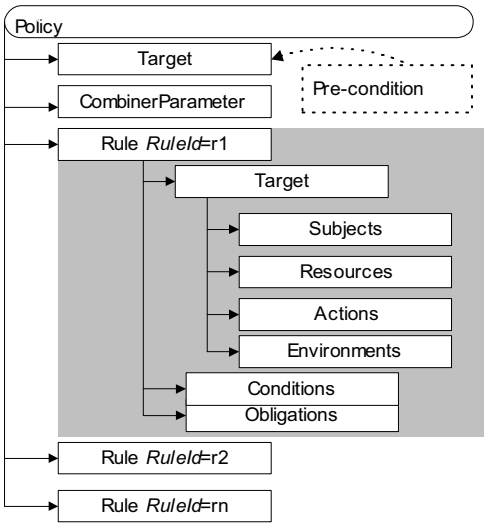

Figure 3. The XACML Policy Container

Policy, from which it can infer the order of disclosure by setting the source and sink nodes [12]. Then trust can progressively be negotiated with a remote party. The policies satisfied during the negotiation phase are eliminated until the sink node is satisfied or the session fails. The trust level and what each party is ready to

give in exchange for his own information is determined by the ordering of the policy set.

4. Discussions

In [13] it was mentioned that policy disclosures are vulnerable to probing attacks. As a result an adversary can use policy disclosure techniques to learn of a party's possession or non-possession of the information being asked for. It is also possible for an attacker to lie by expressing constraints on credentials or services that (s)he does not possess in order to gather information from the attacked. Thus, it is important that both parties receive what they expected and none gains undue advantage [23]. Our exchange protocol uses a gradual, incremental release of information based on a finer trust policy layering that specifies the order in which policies and attributes can be disclosed.

We illustrate our exchange protocol with a simple example illustrated in figure 4. A CIA agent wants to gain access to a CIA web service that is hidden behind a publicly accessible service. Alice, a CIA agent asks for a protected electronic resource res_{CIA}, governed by policy P_{res}, but is unwilling to give up her role attribute certificate $role_{Cert}$ until she is confident that she is communicating with a CIA server. This is specified in ARP_{role}. But P_{res} cannot be disclosed to arbitrary strangers, so it is protected by another policy P. Alice also cannot disclose the access requirement of $role_{Cert}$ - ARP_{role} to arbitrary strangers, so ARP_{role} is governed by another policy ARP_{p1} stating that only US government certified servers can read $ARP_{role.}$ When Alice requests access to res_{CIA}, the server instead of disclosing P_{res}, returns policy P to Alice, which says that P_{res}, can only be disclosed to US government employees ($USGovt_{cert}$).

It is apparent therefore that the first round of policy disclosures is not tightly coupled to resources res_{CIA} and $role_{Cert}$. The assumption is that the kick-off policies cannot explicitly reveal whether both parties possess the required credentials or services. This suggests that both Alice's and the server's behaviour cannot reveal non-possession or possession at the first round of iteration. Again, if the server gives out *USGovtServerCert* and Alice fails to respond with credentials that can satisfy the server's disclosure policy, the negotiation can fail at this point. This is fair: the server has given one of its properties, but neither the access requirements for the sensitive resource nor the resource itself have been disclosed. For trust negotiation to succeed, the policy and credential flow must advance the level of trust, which minimizes the effect of probing attack or lying under false policy expression. This is what makes our approach pragmatic and optimistic. The negotiators must possess a set of credentials (but, of course, this is natural) in order that access control policies can be used to determine the order in which those credentials can be released to advance the trust building session.

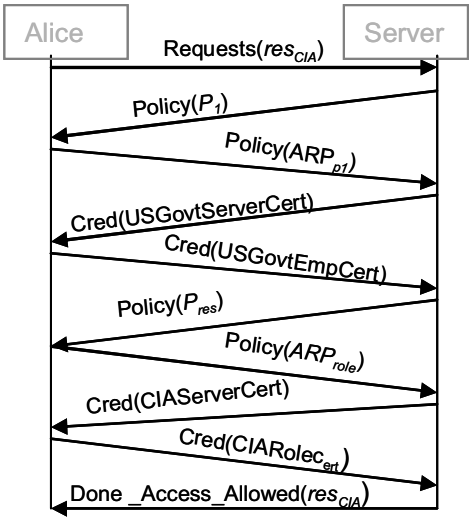

Figure 4. A Simple Negotiation Sequence

5. Conclusions

We have demonstrated how the XACML model can be explored to enable privacy and trust whilst protecting access to electronic resources in a synchronized manner. We described how to construct effective trust policy sets, which can optimize trust establishment sessions, and propose a new trust layer component in the primitive XACML model. We have leveraged trust concepts already proposed by researchers and show how our model optimistically addresses the problem of probing attacks such that the risk to which a party is exposed at any point in the negotiation can be minimized. Our framework has the capabilities to protect resources, policies and credentials simultaneously in distributed environment for users with or without pre-existing trust relationships. The implementation of this framework is in an advanced stage using the SunXACML implementation [17] and the PERMIS Attribute Verifier subsystem.

6. References

[1] A.Anderson, "Privacy Policy Languages: XACML vs EPAL," presented at 5th Annual Privacy & Security Workshop, 2004.

[2] K.E.Seamons, M.Winslett, and T.Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," presented at Network and Distributed System Security Symposium, San Diego, CA, Feb 2001.

[3] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0," http://www.oasis.org, Feb 2005.

[4] M.Lorch, S.Proctor, R.Lepro, D.Kafura, and S.Shah, "First Experience Using XACML for Access Control in Distributed Systems," presented at ACM Workshop on XML Security, Fairfax Va US, 2003.

[5] W.H.Winsborough, K.E.Seamons, and V.E.Jones, "Negotiating Disclosure of Sensitive Credentials," presented at 2nd Conference on Security in Communication Networks, Amlfi, Italy, Sept 1999.

[6] E. F. E.Bertino, A Squicciarini, "TNL: An XML-based Language for Trust Negotiations," presented at IEEE 4th International Workshop on policies for Distributed Systems and Networks, Lake Como Italy, 2003.

[7] A.Acquisti, "Privacy and Security of Personal Information- Economics Incentives and Technological Solutions," presented at Workshop on Economics and Information Security, University of California Berkeley, 2002.

[8] E. Bertino, E.Ferrari, and A. Squicciarini, "Trust Negotiations: Concepts, Systems and Languages," IEEE Computer, pp. 27-34, July/August 2004.

[9] D.W.Chadwick, "The X.509 Privilege Management Infrastructure," presented at Proceedings of the NATO Advanced Networking Workshop on Advanced Security Technologies in Networking, Bled, Slovenia, 2003.

[10] J.Holt and K.E.Seamons, "Interoperable Strategies in Automated Trust Negotiation," presented at 8th ACM Conference on Computer and Communications Security, Philadelphia Pennsylvania, Nov 2001.

[11] W. Winsborough, K. Seamons, and V. Jones, "Negotiating Disclosure of Sensitive Credentials," presented at Second Conference on security in Communication Networks, Amalfi, Italy, September 1999.

[12] T.Barlow, A.Hess, and K.E.Seamons, "Trust Negotiation in Electronic Markets," presented at Eighth Research Symposium in Emerging Electronic Markets, Maastrict Netherlands, Sept 2001.

[13] A.J. Lee, "Traust: A Trust Negotiation Based Authorization Service For Open Systems" Master Thesis, Cornell University, 2003

[14] K.E.Seamons, M.Winslett, T.Yu, B.Smith, E.Child, J.Jacobson, H.Mils, and L.Yu, "Requirements for Policy Languages for Trust Negotiation," presented at 3rd International Workshop on Policies for Distributed Systems and Networks, Moneterey, CA, June 2002.

[15] W. Hommel, "Using XACML for Privacy Control in SAML-Based Identity Federations." presented at "IFIP International Federation for Information Processing CMS 2005 LNCS 3677 pp. 160-169, 2005

[16] S. Nazareth and S. Smith, "Using SPKI/SDSI for Distributed Maintenance of Attribute Release Policies in Shibooleth," Computer Technical Report TR2004-485, 2004.

[17] S. Proctor, "Sun's XACML implementation APIs" http://sunxacml.sourceforge.net/

[18] D.W.Chadwick and O.Otenko, "Implementing Role Based Access Controls Using X.509 Attribute Certificates," IEEE Internet Computing, pp. 62-69, 2003.

[19] D.W.Chadwick and D.P.Mundy, "The Secure Electronic Transfer of Prescriptions," presented at HC2004, Harrogate, UK, March 2004.

[20] D.W.Chadwick and D.P.Mundy, "Policy Based Electronic Transmission of Prescriptions," presented at IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Como Italy, 2003.

[21] S. Brostoff, M. A. Sassea, D. Chadwick, J. Cunningham, U. Mbanaso, and O. Otenko, "RBAC what? Development of a role-based access control policy writing tool for e-Scientists," presented at Workshop on Grid Security Practice and Experience, Oxford UK, 2004.

[22] OASIS, "Security Assertion Markup Language (SAML) Version 2.0," http://www.oasis.org, Feb 2005.

[23] Holger Vogt, Henning Pagnia, Felix Gartner "Modular Fair Exchange Protocols for Electronic Commerce" In Proceedings of the 15th Annual Computer Security Applications Conference, pages 3--11, Phoenix, Arizona, Dec. 1999. IEEE Computer Society Press