Trust Negotiation: Authorization for Virtual Organizations

M. Winslett
Department of Computer Science
University of Illinois
Urbana, IL USA
winslett@uiuc.edu

Adam J. Lee
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA USA
adamlee@cs.pitt.edu

Kenneth J. Perano
Sandia National Laboratories
Livermore, CA USA
perano@ca.sandia.gov

ABSTRACT
Trust negotiation is an authorization approach for open distributed systems, such as dynamic coalitions and other types of virtual organizations. Under the trust negotiation approach to authorization, every resource that might be shared within the coalition is protected by an access policy that describes the attributes of those qualified to access it (e.g., employer, job title, role, age). Each party collects digital credentials, such as X.509 attribute certificates or SAML assertions, from credential issuers who can attest to that party’s attributes. At run time, a resource owner and potential client exchange information on their access policies and attributes, to determine whether the client possesses the attributes necessary to gain access, and vice versa. Trust negotiation has a firm theoretical foundation and a number of freely available implementations.

In this paper, we argue that trust negotiation is ready for a trial deployment in a real-world application. We describe the software available for a deployment, including the flexible TrustBuilder2 framework for experimenting with trust negotiation runtime systems, and the CLOUSEAU compliance checker, which can quickly determine whether a set of credentials complies with a particular policy. We also describe the Traust approach for letting legacy applications take advantage of trust negotiation.

Categories and Subject Descriptors: K.6.5
[Management of Computing and Information Systems]: Security and Protection

General Terms: Security

1. INTRODUCTION
In large-scale open systems, resource providers allow their resources to be shared across organizational boundaries. Example open systems include supply chains, coalitions of first responders, military coalitions, grid computing, and public web services and their clients. Due to the large number of potential users in these environments, we cannot assume that resource providers will know in advance the identities of all qualified users who might possibly wish to access their resources. Even if the set of potential users is known, the administrative overhead of maintaining the user list is unappealing, as internal changes in partner organizations must be tracked to keep the list up to date. In addition, we cannot assume that clients already are familiar with the resource providers that they may wish to interact with. This suggests several requirements for an authorization system designed for use in open computing systems.

Bilateral trust establishment. To enable effective resource sharing, clients and resource providers must be able to establish trust in each other at run time, rather than relying on a pre-existing trust relationship.

Discovery of access policies at run time. In large-scale open systems, often a client will not be familiar with the authorization policy of a resource that she wants to access. The client should be able to discover this policy on the fly.

Respect for sensitive information. To protect clients and resource providers from malicious entities, their interactions should reveal no more information than necessary. Clients and resource providers should have some ability to control their disclosure of sensitive information, including their objectives, policies, identities, and attributes.

Scalability. The system must scale well in performance and administrative overhead as the set of users and resources grows.

Flexibility. Because each virtual organization is different, the authorization system should be easy to adapt to new authorization requirements.

Support for legacy systems. Adding trust negotiation to existing open systems should not require a complete redesign of deployed applications, protocols, or the network infrastructure.

To address these needs, researchers have developed the theoretical foundation for trust negotiation, together with a number of prototype implementations. Progress to date has been substantial, to the point where we believe that trust negotiation is ready for a trial run with a real application from an early adopter. In this paper, we introduce trust negotiation in Section 2, and present the TrustBuilder2 framework for trust negotiation in Section 3. We describe the CLOUSEAU policy compliance checker in Section 4, and the Traust approach for handling legacy applications in Section 5. Section 6 concludes the paper.

2. BACKGROUND & RELATED WORK
In trust negotiation, the access policy for a resource is a declarative specification of the attributes that a party must possess.
to access the resource. Trusted parties issue digital credentials that certify these attributes. For example, an employee might have a digital employee ID card issued by his company. These credentials are also considered resources, so credentials that contain sensitive information can be protected by disclosure policies of their own. In this way, an access request leads to a bilateral and iterative disclosure of credentials and policies between the user and resource provider. Trust is established incrementally, as more and more sensitive credentials are disclosed between the user and resource provider.

For example, consider the negotiation in Fig. 1. Here, Alice has attempted to access a service owned by Bob. Her access attempt triggers a trust negotiation, which is automatically carried out by two small software agents acting on their behalf (labeled “Alice” and “Bob” in Fig. 1). After a message exchange where they set up the parameters of the negotiation (e.g., credential formats supported and the version of the protocol that they are running), Bob sends Alice a copy of the access policy for the service she has requested. The policy says that only VISA employees can gain access. Alice is a VISA employee, but she considers that information sensitive. She is only willing to show her employee ID to members of the BBB. She sends (“discloses”) that policy to Bob, who responds by sending her his BBB credential, along with a proof that he owns it. Bob’s BBB credential satisfies Alice’s policy for her ACME card, so she sends her ACME credential to Bob, along with a proof that she owns it. Then Bob grants Alice access to the service.

As a more challenging scenario, consider high-assurance environments such as disaster management networks, or critical infrastructures such as the electric power grid. In these environments, qualified individuals must be able to gain access to resources quickly. For instance, during a crisis, responding police, fire, and rescue workers must be able to access status information about the crisis, even if they are from another jurisdiction. In some situations in the electric power grid, market-sensitive information should be released to competitors to help them react appropriately to changing conditions in the grid and avoid cascading blackouts. More generally, the following additional authorization system features are relevant:

- Emergencies are usually highly context-dependent, which may make it hard for users to understand why their access requests are permitted or denied. On request, the trust negotiation system must provide explanations for the success or failure of users’ access requests.

- Often critical decisions must be made quickly. Users and resource providers will not tolerate long delays when requesting access. Denial-of-service attacks must be stymied.

- Under emergency conditions, resource providers may take a more permissive approach to sharing, in hopes of solving critical problems. Thus users will likely act in uncharacteristic ways during times of crisis, and these atypical activities may be investigated after the crisis is past. Users and resource providers will need verifiable audit trails of activities.

For a large open distributed system, one should not assume that all participants run identical trust negotiation software and algorithms. Theoretical results in the literature ensure that to a large extent, diversity can be accommodated without compromising the ability of the negotiation to result in the client’s gaining access to the target resource, and guaranteeing that trust negotiations always terminate [21]. Researchers have devised explanation facilities to help users understand why a negotiation had a particular outcome [6], and analysis algorithms to help administrators understand the effects of potential changes in large sets of policies [20].

Several implementations of trust negotiation systems have also been described in the literature, including TrustBuilder [12] and TrustBuilder2 [15], Trust-X [6], the system of Koshytanski and Massacci [13], the system of De Coi and Olmedilla [9] (which in turn incorporates many aspects of the PeerTrust [19] and PROTUNE [7] trust negotiation systems), and TuLiP [8]. Among these, TrustBuilder2 differs in having been intended to serve as a general platform for experimenting with trust negotiation, as described more detail in Section 3.

To decide whether a negotiation has succeeded, and to determine what action to take next if it has not, a participant’s agent must determine which subset(s), if any, of a set of credentials satisfy a particular policy. This activity is potentially quite expensive, and it is performed repeatedly during a trust negotiation. Traditionally, this policy compliance test required the use of theorem-proving techniques, and theorem-provers are quite slow. In Section 4, we briefly describe an alternative approach to policy compliance checking that is much faster and lighter-weight than a traditional theorem-prover, and is available as part of the TrustBuilder2 release.

3. TRUSTBUILDER2 FRAMEWORK
To encourage systems experimentation with trust negotiation, we have released TrustBuilder2, a fully configurable and extensible framework for prototyping and evaluating trust negotiation systems [14, 15]. TrustBuilder2 employs a plug-in based architecture, extensible data type hierarchy, and flexible communication protocol to provide a framework within which numerous trust negotiation protocols and system configurations can be quantitatively analyzed. Since its initial release on the web in the summer of 2007, TrustBuilder2 has been downloaded over 300 times. We have also used TrustBuilder2 as the basis for our own experiments with trust negotiation, focusing on performance and extensibility. At this point, we believe that the technology is
ready for adoption in a trial deployment, the results of which will identify the problems that our future research should address. We have identified two emergency response applications at Sandia National Laboratories as candidates, and we are looking for others, as well as for funding for a trial.

Almost every component of the TrustBuilder2 framework can either be extended or replaced by a user-defined plug-in. Because the TrustBuilder2 framework was developed using Java, plug-ins can be loaded dynamically at runtime without any modification to the TrustBuilder2 framework itself. Extensions to the primary components of TrustBuilder2, as well as plug-ins that interpose between these components, can be added to the system by creating classes conforming to the appropriate interfaces.

We evaluated the performance of TrustBuilder2 in conjunction with the CLOUSEAU compliance checker discussed in the next section. As detailed in [14], for a negotiation requiring one round of messages for setup, followed by three round-trip exchanges of credentials and policies, the average time to conduct the negotiation was 434.73 ms (standard deviation 97.56 ms), exclusive of network costs. The first trust negotiation was three times longer than the average (1350 ms), due to Java startup costs.

4. CLOUSEAU COMPLIANCE CHECKER

Checking compliance with trust management policies is one of the most expensive portions of the trust negotiation process. To ensure that a trust negotiation session succeeds whenever possible, authorization policy compliance checkers must be able to find all minimal sets of their owners’ credentials that can be used to satisfy a given policy. Unfortunately, theorem provers are quite slow at carrying these tasks.

We built the CLOUSEAU compliance checker to address this problem [16]. Using a provably-correct procedure, CLOUSEAU compiles policies into a low-level language for internal use, then leverages an efficient Rete [10] pattern-matching algorithm to find all satisfying sets of credentials for a given policy in time that grows as $O(N^3A)$, where $N$ is the number of satisfying sets for the policy and $A$ is the average size of each satisfying set. In practice, CLOUSEAU is much faster than a theorem-prover. To date, we have built compilers for policies in the RT and WS-SecurityPolicy languages. Since all existing trust negotiation policy languages rely on the same kinds of evidence, we expect that compilers can be built for other languages as well.

CLOUSEAU requires only tens of milliseconds, on average, to determine every satisfying set of credentials associated with a reasonably-sized policy; this is comparable to the time required by previous trust negotiation compliance checkers to find one satisfying set. In the worst case, the number of satisfying sets for a given policy can be exponential in the size of the policy. However, CLOUSEAU’s performance remains reasonable even when policies become inordinately complex. Even policies as complex as even the most complicated policies used in Becker’s health records service [1] can be analyzed by CLOUSEAU in under 100 ms [14].

5. LEGACY APPLICATIONS

To allow legacy applications to take advantage of trust negotiation without major modifications, we proposed Trust [17], a stand-alone authorization service that allows the adoption of trust negotiation in a modular, incremental, and grassroots
manner, without application software or protocol upgrades. Traust servers act as brokers for the security tokens needed to gain access to shared resources. These tokens can be of any format, including (username, password) pairs, Kerberos tickets, SAML assertions, and X.509 certificates. Clients contact Traust servers and negotiate for access tokens for logical or physical resources, including network servers, RPC methods, and organization-wide roles. The Traust service also provides clients in the system with an opportunity to establish trust in the service prior to the disclosure of their (potentially sensitive) resource access requests.

6. CONCLUSIONS

In this paper, we have discussed TrustBuilder2, CLOUSEAU, and Traust, which together offer a new approach to authorization in open distributed systems. TrustBuilder2 is a flexible and reconfigurable Java-based framework for experimenting with trust negotiation. TrustBuilder2 supports a plug-in based architecture to allow any system component to be modified or replaced by users of the system, without requiring modification or recompilation of the underlying framework. TrustBuilder2 is also agnostic with respect to the formats of credentials and policies used during the negotiation. CLOUSEAU complements TrustBuilder2 by providing fast compliance checking for access policies, without the need for a theorem-prover. Traust suggests a potential means of allowing legacy applications to take advantage of trust negotiation.

Released at http://dais.cs.uiuc.edu/dais/security/tb2 during summer 2007, TrustBuilder2 has been downloaded over 300 times. Based on our positive experience to date, we are actively seeking collaborators and funding for a trial deployment of TrustBuilder2. Feedback from a real-world test will allow us to identify and address whatever remaining research issues may hinder future deployment of trust negotiation in open systems.

7. ACKNOWLEDGMENTS

The creation of TrustBuilder2, CLOUSEAU, Traust, and their underlying theoretical foundations was supported by Motorola, NSF under grants IIS-0331707, CNS-0325951, and CNS-0524695, and Sandia under DOE SNL 541065.

8. REFERENCES