SAND2006-3046C

# Traust: A Trust Negotiation Based Authorization Service

Adam J. Lee[1], Marianne Winslett[1], Jim Basney[2], and Von Welch[2]

[1] Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{adamlee, winslett}@cs.uiuc.edu

[2] National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{jbasney, vwelch}@ncsa.uiuc.edu

**Abstract.** In this demonstration, we present Traust, a flexible authorization service for open systems. Traust uses the technique of trust negotiation to map globally meaningful assertions regarding a previously unknown client into security tokens that are meaningful to resources deployed in the Traust service's security domain. This system helps preserve the privacy of both users and the service, while at the same time automating interactions between security domains that would previously have required human intervention (e.g., the establishment of local accounts). We will demonstrate how the Traust service enables the use of trust negotiation to broker access to resources in open systems without requiring changes to protocol standards or applications software.

## 1 Introduction

Making intelligent authorization decisions in large computer systems is a non-trivial task. Traditional authorization systems require some explicit notion of the users accessing the resources provided by the system; this knowledge is usually in the form of a user account protected by a password or some other digital credential. While systems such as Kerberos [5] and hierarchical PKIs [4] help reduce the overhead of managing these systems on a per-organization basis, it is widely accepted that they do not scale well in the context of large-scale open systems where information and resources are shared across organizational boundaries.

The increasing popularity of the Internet has led to a surge in the number of resources provided through open environments such as the world wide web, peer-to-peer networks, virtual organizations, disaster response networks, joint task forces, and grid computing environments. In these systems, it is unreasonable to assume that entities will have—or even need to have—explicit knowledge of the peers that they are communicating with. For instance, can users of a peer-to-peer network reasonably be expected to enforce access controls on their shared

resources based on the identities of the thousands of other peers in the system? We argue that in the context of large-scale open systems, authorization decisions are best made based on the attributes of the users in the system, as this allows for better scalability as the number of users continues to increase.

Trust negotiation [10] is a technique developed to allow peers to conduct bilateral and iterative exchanges of digital credentials to bootstrap trust relationships in open systems. To date, current work in trust negotiation has focused on the development of languages and strategies for trust negotiation [6, 7, 12], or the embedding of trust negotiation into commonly used protocols [3, 9]. In fact, little attention has been focused on designing a general-purpose authorization system based on trust negotiation. In this demonstration, we present Traust, a general purpose authorization service based on trust negotiation. Traust provides a uniform interface for clients to locate and obtain the credentials necessary to access resources provided by systems in a different security domain and acts as a viable migration path for the adoption of trust negotiation research into existing open systems.

## 2 Goals

The design of Traust embodies five major design goals. These goals build on the strengths of trust negotiation techniques developed in the past and help Traust act as a scalable and flexible authorization service for large-scale open systems.

**Bilateral trust establishment** It is important not only for a service provider to trust the clients requesting its services, but for clients to trust the services that they choose to interact with. Before disclosing any requests or credentials to the Traust service, clients have the opportunity to conduct a content-triggered trust negotiation session [2] to protect their potentially sensitive requests.

**Run time access policy discovery** In large-scale open systems, clients cannot be expected to know the access requirements of services of interest a priori. Traust supports the dynamic discovery of these policies as part of the authorization process.

**Privacy preserving** An interaction between a client and the Traust service should not reveal any extraneous information about either the client or the service. Trust should be established iteratively, each entity providing more sensitive credentials in response to the disclosures of the other entity.

**Support for legacy and trust-aware applications** Incorporating the Traust service into existing open systems should not involve completely redesigning deployed applications, protocols, or network infrastructure. Traust should support tight interaction with trust-aware applications via the use of a public API, but also remain accessible to clients who wish to access legacy applications.

**Light-weight, yet robust** The Traust service should be light-weight enough for a single user (e.g., a peer-to-peer client) to deploy on her local machine, yet robust enough to meet the demands of a large security domain.

## 3 Design and Implementation

The Traust service was designed to provide a mechanism through which trust negotiation can bridge the security gap that exists between security domains in large-scale open systems without requiring widespread protocol or application software updates. Traust servers act as authorization brokers that distribute access tokens for certain services deployed within their security domain to *qualified* outsiders. Traust client software allows users to carry out a number of trust negotiations with the Traust server; these negotiations allow both the client and server to establish some degree of trust in one another.

Traust relies on SSL to protect the confidentiality and integrity of connections between clients and the service. Upon connecting to the Traust service, clients have the opportunity to conduct a content-triggered trust negotiation with the service to gain some level of trust before disclosing a potentially sensitive resource request. If this negotiation succeeds, the client then discloses its resource request to the Traust server. Resources can be as specific as a particular RPC method call or as broad as a request for access to a system-wide role. When the Traust service receives such a request, it locates the policies protecting the requested resource and initiates a trust negotiation with the client to determine whether the client is a qualified outsider. If this negotiation succeeds, the Traust service issues the client any credentials needed to access the requested service. The server can obtain these tokens through either accessing static tokens in its credential repository, referencing files in its file system, or interfacing with external processes (e.g., one-time password generators, Kerberos servers, or MyProxy servers [8]). There are no fundamental limits on the types of credentials that can be issued.

Our implementation of the Traust service is written in Java and leverages the TrustBuilder framework and protocol for trust negotiation [11]. TrustBuilder has been successfully incorporated into several protocols [3, 9] and currently supports the use of X.509 attribute certificates as its native form of credentials and the IBM Trust Establishment language [1] for trust negotiation. Our implementation of the Traust service currently supports the issuance of username/password pairs and X.509 proxy certificates. We have developed both a stand-alone client application that can be used to manually obtain credentials to access legacy services and a client API that can be incorporated into the design of newer, trust-aware applications.

## 4 Demonstration

In our demonstration, we show how the Traust service can enable the use of trust negotiation to grant qualified users access to a legacy resource without requiring any changes to the underlying resource or applications used. We illustrate how a volunteer rescue dog handler can use Traust to gain access to a web-based information portal used to coordinate the recovery effort for an earthquake, despite having no pre-existing trust relationship with the portal. The user first browses

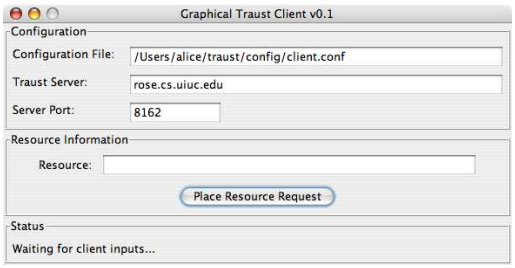


**Fig. 1.** Screenshot of the graphical Traust client.

to this web site and is presented with a login form and resource descriptor to pass into her Traust client. She then uses our graphical Traust client to initiate an interaction with the Traust server responsible for protecting access to this site (see Fig. 1). This interaction allows the client to establish trust in the server (by verifying that the server can demonstrate proof-of-ownership of a state-issued disaster response coordinator credential) and allows the server to gain trust in the user (by verifying that she can demonstrate proof-of-ownership of trusted credentials which indicate that he is a certified rescue dog handler with up-to-date vaccinations). The Traust server then returns a one-time-use password for the web site. Further information regarding our demonstration can be found at `http://dais.cs.uiuc.edu/~adamlee/research/traust/demo/disaster_demo.html`.

## References


[1] A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *IEEE Symposium on Security and Privacy*, May 2000.

[2] A. Hess, J. Holt, J. Jacobson, and K. E. Seamons. Content-triggered trust negotiation. *ACM Transactions on Information System Security*, 7(3), Aug. 2004.

[3] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons, and B. Smith. Advanced client/server authentication in TLS. In *Network and Distributed Systems Security Symposium*, Feb. 2002.

[4] R. Housely, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. RFC 2459, Jan. 1999.

[5] J. Kohl and C. Neuman. The Kerberos network authentication service (V5). RFC 1510, Sep. 1993.

[6] H. Koshutanski and F. Massacci. Interactive trust management and negotiation scheme. In *2nd International Workshop on Formal Aspects in Security and Trust (FAST)*, pages 139–152, Aug. 2004.

[7] N. Li and J. Mitchell. RT: A role-based trust-management framework. In *Third DARPA Information Survivability Conference and Exposition*, Apr. 2003.

[8] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: MyProxy. In *Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, Aug. 2001.

[9] T. van der Horst, T. Sundelin, K. E. Seamons, and C. D. Knutson. Mobile trust negotiation: Authentication and authorization in dynamic mobile networks. In *Eigth IFIP Conference on Communications and Multimedia Security*, Sep. 2004.

[10] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, Jan. 2000.

[11] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. The TrustBuilder architecture for trust negotiation. *IEEE Internet Computing*, 6(6):30–37, Nov./Dec. 2002.

[12] T. Yu, M. Winslett, and K. E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1), Feb. 2003.