

A Privacy Preserving Enhanced Trust Building Mechanism for Web Services

Zhengping Wu, Alfred C. Weaver

Department of Computer Science, University of Virginia

151 Engineer's Way, P.O. Box 400740, Charlottesville, Virginia 22904-4740

{zw4j, acw}@cs.virginia.edu

Abstract

With the development of web services, more effective trust building mechanisms are needed to deploy diverse trust models in a web services environment. The lack of mechanisms that can dynamically build trust relationships while preserving privacy impedes progress. Current web service technologies encourage a client to reveal all its private attributes in a pre-packaged digital credential to the service provider as a requirement for verification. This leads to privacy leakage. We propose a mechanism that discloses only the attributes required to negotiate a trust relationship, thereby preserving the client's privacy. After the trust negotiation is successful and trust has been established, we use dynamic validation to monitor and maintain the trust relationship.

Keywords: Trust building, privacy preserving, web services

1. Introduction

Nowadays, web services provide crucial support for online business. The client of a web service and the service provider often share no prior relationship and no common security domain. While some web services permit anonymous access to low-sensitivity information such as weather reports, most require strong user authentication before allowing access to sensitive information such as electronic medical records protected by HIPAA [1]. To access this information, the client must first build a trust relationship with the services. Daily life provides many examples of the need for trust relationships before any access privilege is granted. For instance, governments insist upon proof of citizenship before issuing a passport; banks must verify a person's identity before opening an account; university libraries need to verify student status before lending books. In these examples, a client first initiates a request; then the service provider responds with a request for specific attribute information required by that organization's policies; the client either provides the attributes required, in which case the request is usually granted, or the client fails to do so, and the request is denied. Web services operate in an analogous way. If a student (client) attempts to open a free student checking account using online services, the service provider must verify the client's eligibility and the client must supply an acceptable

security token that confirms the client's student status. A security token is a collection of security-related information that is conveyed within a SOAP message [2]. Only then will the service provider build a trust relationship with the student and grant the privileges needed to open a new account. The exchange of credentials in paper documents or security tokens can lead to information leakage. First, a credential may not be used for its intended primary purpose. For example, a driver's license is often used as a proof of age when its intended function is to document permission to operate a vehicle. Second, a pre-packaged credential may reveal more information than is necessary. For example, a student ID may reveal the holder's degree information, an attribute not required by the bank to establish student status.

Another common way of building a trust relationship within web service environments is for the service provider to require pre-enrollment of its clients before granting any access privileges. As discussed in Ferraiolo et al.'s paper [4], pre-enrollment and conditioning of all future access upon the satisfactory presentation of tokens that contain role information is required for all clients. However, this approach is contrary to the basic philosophy of web services, which anticipates choosing services dynamically at run-time.

In this paper we propose a mechanism that reveals the minimal number of attributes necessary to build the desired trust relationship for the web service environment. After that, any changes of policy requirements associated with this new trust relationship are dynamically enforced using a trust group element.

2. Trust Building Related Facilities

Web services use the Simple Object Access Protocol (SOAP) to exchange information. EXtensible Markup Language (XML) is used with SOAP to define an extensible messaging framework that can exchange a message over a variety of underlying protocols. Neither SOAP nor XML provides any security mechanisms for the information exchanged. Web service security is based on a process in which a service provider requires that an incoming access request prove a set of claims such as name, public key, or permission. The service provider indicates its required claims in its policy document.

More recently, the WS-Trust specification [7] enables applications to construct trusted message exchanges and

provides a flexible set of mechanisms that can be used to support a range of security protocols. Here, a security token service (STS) is a web service that issues security tokens [2,7]. STS makes assertions based on evidence that it trusts. At the same time, the private attributes of principals within a trust domain are maintained by an attribute service. Trust establishment is the mechanism by which one entity relies upon a second entity to execute a set of actions or make a set of assertions. But this specification does not mention how to build and represent trust relationships. The trust relationships conveyed in these actions or assertions can be built through exchanging security tokens. Thus the token-based mechanism provides an applicable solution to build trust relationships in a dynamic environment of web services.

3. Trust Primitive Element

Definition 1: A trust primitive (TP) is defined as the minimal subset of attributes in a pre-packaged digital credential which has a complete semantic meaning according to a set of policy requirements. A trust primitive is signed by the credential holder, which is either an individual user or a security domain.

In the proposed privacy preserving enhanced trust building mechanism, a trust primitive is represented as a subset of attributes in the attribute service and conveyed as an XML element when it is exchanged between security domains. As illustrated in figure 1, trust primitive 1 corresponds to a library access rule with three required attributes (4, 6, and 7). The holder of the digital credential forms this trust primitive and will be allowed to use the electronic library if the server can verify that its three attributes are valid. For library checkout, trust primitive 2 contains the same three attributes in trust primitive 1 plus attribute 2, which is needed for accounting purpose. Trust primitive 3, consisting of attributes 3, 6 and 7, might be used to verify same-sex gender before granting entrance to a dorm floor.

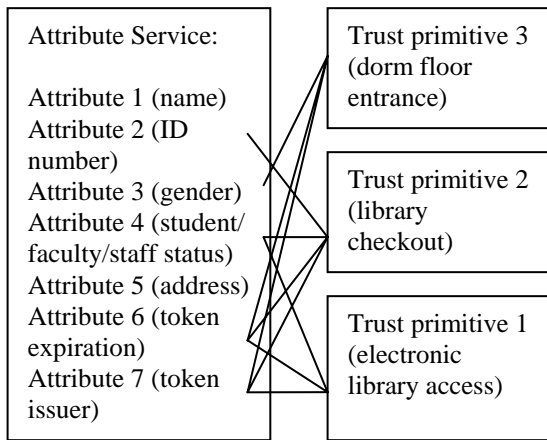


Figure 1. Attributes and three trust primitives

Every request for retrieving a trust primitive defined by the credential holder from an outside domain will be verified by the attribute service to selectively disclose the subset of attributes associated with the trust primitive. Another merit of trust primitives is that they prevent initiation of selective disclosure from anyone except the credential holder. Figure 2 shows the proposed protocol for privacy preserving during the trust building process. A client and a service provider from different security domains are assumed. Before the beginning of this privacy preserving enhanced trust building process, the attributes of the client are maintained by an attribute service. The client is also assumed to hold a security token containing its identity information. The protocol is initiated by a need for the client to disclose some of its attributes to the service provider for negotiation. Then the protocol executes steps one through ten to complete a round of negotiation.

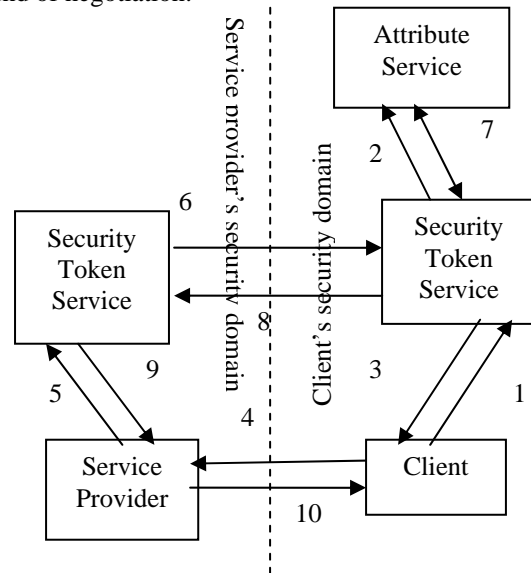


Figure 2. Privacy Preserving Protocol for Trust Building Process

- (1) The client initiates a request to register a TP corresponding to the policy from the service provider.
- (2) The client's STS registers the TP in the attribute service.
- (3) The STS adds the TP to the client's security token.
- (4) The client embeds the security token and sends the request.
- (5) The service provider asks its own STS to check whether the request complies with the service's policy.
- (6) Provider's STS performs verification using the embedded TP.
- (7) Client's STS uses this TP as the keyword to retrieve the corresponding attributes.
- (8) Client's STS returns the attributes.
- (9) Service provider's STS finishes verification and returns its decision (grant/deny).
- (10) The service provider either performs the requested operation or issues a denial.

Sometimes the client also needs to verify some of the service provider's attributes to negotiate a trust relationship, so the roles of the client and the service provider can be interchanged. Several rounds of exchange form the negotiation needed to build a trust relationship.

4. Trust Group Element

Definition 2: A trust group represents a collection of partners who comply with the same set of policy requirements. The partners are entities who have direct trust relationships with the policy holder.

A trust group name is associated with a set of policy requirements. For example, if the service provider creates a set of policy requirements in the form of a WS-Policy file for negotiation of a trust relationship, the trust group name will be attached at the end of the file. Every partner complying with this set of policy requirements will use this trust group name to represent the corresponding trust relationship.

After negotiation, a new trust group element is added to the client's security token, and the security token is signed by the client's STS again. Alternatively, the security token is replaced by a new security token issued by the service provider's STS, which contains the corresponding trust group element representing the new trust relationship. A trust group element is represented by three XML tags (trust group name and two domain/individual identities). Every policy holder also needs to record all the trust group elements in which the holder is involved; the same is true for all partners.

After trust is established, subsequent requests are validated by trust group elements in security tokens. If the access requirement for a trust group element changes in a service provider's policy, then the service provider needs to revalidate the previous trust relationship by invalidating the old trust group element, asking for the client's trust primitives again, and then verifying whether the new trust primitives meet the requirements of the changed policy. If access is granted, a new trust group element will replace the old one for future use.

5. Implementation

The architecture of our privacy preserving enhanced trust building mechanism is illustrated in figure 3. The web service client and provider build a dynamic trust relationship via negotiation engines and security token services in both security domains. At the same time, the security token service uses an attribute service to register trust primitives for the client. Negotiation engines control the overall workflow to build trust relationships dynamically. We implemented this system architecture on the Microsoft .Net platform with the Web Service Enhancement (WSE) toolkit. All the building blocks in our system use web services as interfaces. We also use a

graphic user interface to help users define and sign their trust primitives.

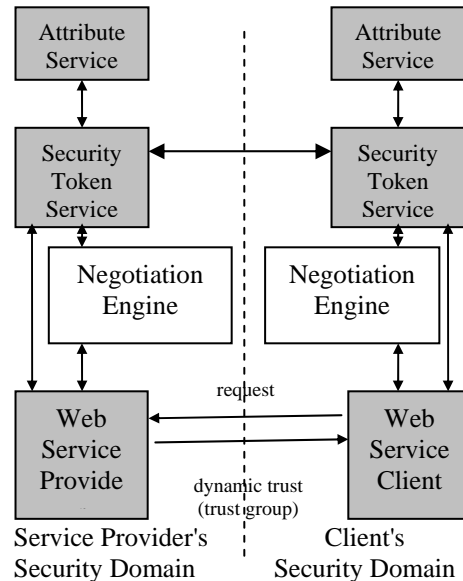


Figure 3. Diagram of the system architecture

We have implemented this privacy preserving enhanced trust building mechanism in a federated cyber trust system [9] for a healthcare environment. It provides the functionality of negotiating and building trust relationships for the whole system.

6. Discussion

At this moment there are no standard criteria for evaluating trust building mechanisms. Seamons et al. [3] proposes some requirements for policy languages for trust negotiation. Some of the requirements for expressiveness and semantics are also important in trust building, such as monotonicity and credential combinations. Disclosure of additional attributes or credentials in our trust building mechanism guarantees no reduction of privileges, because additional attribute disclosure corresponds to satisfying a new policy or fulfilling an additional requirement for additional privileges; monotonicity is thus guaranteed. Our trust building mechanism provides selective disclosure of attributes, so arbitrary credential combinations can be easily achieved.

With regard to the trust architecture proposed by IBM/Microsoft in [2], that scheme provides no protection in either the attribute service or the security token service that would allow the client to disclose its attributes selectively. In contrast, our mechanism better respects clients' privacy.

7. Related Work

Several types of trust building mechanisms have been described in the literature. The most basic way is to map the identity of the client (or one identity in the client

security domain) to one identity in the service provider security domain. Some other extant trust management systems build trust relationships between different security domains using the client's role as a basis for mapping. More recently, group-based mechanisms have been proposed to build trust relationships.

In a complex organizational setting such as a healthcare environment one might assign differing roles, and hence differing access permissions, to physicians, technicians, and patients. In Sandhu et al.'s paper on role-based access control [10], a role-based trust building process is implied by setting a mapping between local roles and roles within remote domains, which we call a role-to-role mapping. In the mechanism proposed by Freudenthal et al. [11], predefined trust relationships are used to complete the trust building process. The common approach of [10] and [11] is that the authors try to decide whether to grant access at run-time by deciding what permissions each client has according to the assigned role and the predefined trust relationship associated with that role. The use of role as a basis for trust building creates a problem in domain-to-domain interactions, namely the potential misalignment of the precise definition of roles from one domain to another. The consequence could conflict with the principle of least privilege, which limits the security privileges to actual needs.

Vandenwauver et al. [6] and Van Dyke [5] both use group-based mechanisms to describe a collection of security requirements agreed to by the administrators for a group of domains. Li et al. [8] proposes an entire trust management framework that can group logically related objects so that permissions about them can be assigned in one operation. All the group-based mechanisms mentioned above require some superior authority to create or predefine group information, and thus the resulting trust building mechanisms are not fully dynamic and lack of negotiation.

The most important point is that all these identity-based, role-based and group-based mechanisms do not address privacy issues, which could lead to information leakage during the trust building process.

8. Conclusion

In this paper we described a privacy preserving enhanced trust building mechanism that extends the extant trust building mechanisms for web services to gain many advantages from its privacy control and dynamic capabilities. Our research motivation comes from the complicated privacy requirements inherent to current healthcare data management and similar sensitive information management. Our new trust building mechanism is dynamic with these advantages:

- It allows only the client to choose what attributes may be viewed by the service provider. Therefore, it is capable of enforcing the client's privacy.

- It allows only the chosen attributes to be viewed by the service provider. Therefore, it is capable of disclosing private attributes selectively.
- It allows any trust relationship to be renewed whenever the service provider's policy is updated. Therefore, it is inherently dynamic.

9. References

- [1] Health Care Portability and Accountability Act, Public Law 104-191, <http://aspe.hhs.gov/admsimp/pl104191.htm>, August 1996.
- [2] A Joint White Paper from IBM Corporation and Microsoft Corporation, "Security in a Web Services World: A Proposed Architecture and Roadmap", <http://msdn.microsoft.com/library/en-us/dnwssecur/html/securitywhitepaper.asp>, April 2002.
- [3] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu, "Requirements for Policy Languages for Trust Negotiation", Proceedings of 3rd International Workshop on Policies for Distributed Systems and Networks, Monterey, California, June 2002.
- [4] D. Ferraiolo et al. "Proposed NIST Standard for Role-Based Access Controls". ACM Trans. Information and System Security, August 2001, 4(3) pp. 224–274.
- [5] James Van Dyke. "Establishing Federated Trust Networks among Web Services", B. S. thesis, University of Virginia, March 2004.
- [6] M. Vandenwauver, R. Govaerts, J. Vandewalle, "Role based access control in distributed systems", Communications and Multimedia Security, 1997 volume 3, pp. 169-177.
- [7] Steve Anderson, et al., "Web Services Trust Language (WS-Trust)", <http://msdn.microsoft.com/ws/2005/05/ws-trust/>, February 2005.
- [8] Ninghui Li, John C. Mitchell, William H. Winsborough. "Design of a Role-Based Trust-Management Framework", Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 114-130.
- [9] AC Weaver, et al., "Federated, Secure Trust Networks for Distributed Healthcare IT Services", Proceedings of IEEE International Conference on Industrial Informatics, August 2003, pp. 162-169.
- [10] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-based access control models", IEEE Computer, 1996, 20(2), pp. 38-47.
- [11] Eric Freudenthal, et al., "dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments", Proceedings of 22nd International Conference on Distributed Computing Systems, 2002, pp. 411-420.