A pattern for the WS-Trust standard for web services

Ola Ajaj and Eduardo B. Fernandez
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431-0991 USA
oajaj@fau.edu, ed@cse.fau.edu

Abstract: Web services intend to provide an application integration technology that can be successfully used over the Internet in a secure, interoperable and trusted manner. One of the main functionalities of web services is providing secure messaging, where the web services exchange security credentials (either directly or indirectly). However, each party needs to determine if they can trust the asserted credentials of the other party. Moreover, the dynamic interaction between the web services requires specifying trust relationships in an explicit way for all parties. Without a clear definition of how web services could manage secure communications and establish trust relationships with other partners, malicious web services could use their business interactions to perform illegal actions. The WS-Trust standard defines how to establish trust between interacting parties; we present here a pattern for this standard. WS-Trust defines a security token service and a trust engine which are used by web services to authenticate other web services. Using the functions defined in WS-Trust, applications can engage in secure communication after establishing trust.

1. Introduction

Without a clear definition of how web services can manage secure communications and establish trust relationships with other partners, it would be hard to perform any kind of interaction. WS-Trust is a standard to support the establishment of trust relationships.

Using web services requires that we exchange credentials to define the rights of each participant. This exchange is based on trust and builds further trust. Trust is based on security and other policies to enable requesting and obtaining credentials within different trust domains. Both parties need to determine if they can "trust" the asserted credentials of the other party. The goal of the WS-Trust standard is to enable applications to construct trusted message exchanges. This trust is realized through the exchange and brokering of security tokens [oas09].

The motivation toward WS-Trust is supported by the fact that there are different formats for security tokens (e.g. X.509 certificates, Kerberos tickets, SAML assertions, XACML policies, etc.), and it's unlikely to expect that an endpoint will understand each of these options. Additionally, there is no guarantee that there will be an intersection between the sets of supported security token formats of different actors who are willing to exchange messages using the WS-Security standard [Mad03].

Web services standards are rather complex and verbose and it is not easy for designers and users to understand their key points. By expressing web services security mechanisms and standards as patterns, we can verify if an existing product implementing a given security mechanism supports some specific standard [Fer06]. Inversely, a product vendor can use the standards to guide the development of the product. By expressing standards as patterns, we can compare them and understand them better. For example, we can discover overlapping and inconsistent aspects between them. We have produced

patterns to describe SAML, XACML, WS-Policy, WS-Security, XML Encryption, XML Digital Signature, and others. A standard defines a generic architecture and this is a basic feature of any pattern; it can then be confirmed as a best practice by looking at products that implement the standard (and implicitly the pattern).

Section 2 shows a pattern that describes this standard. Section 3 ends the paper with some conclusions.

2. A Pattern for WS-Trust

Intent

WS-Trust defines a security token service and a trust engine which are used by web services to authenticate other web services. Using the functions defined in WS-Trust, applications can engage in secure communication after establishing trust.

Example

The *Ajiad* travel agency offers its travel services through several different business portals to provide travel tickets, hotel and car rental services to its customers. *Ajiad* needs to establish trust relationships with its partners through these portals.

The *Ajiad* supports different business relationships and needs to be able to determine which travel services to invoke for which customer. Without a well-defined structure, *Ajiad* will not be able to know if a partner is trusted or not, or to automate the trust relationships quickly and securely with its partners, which may lead to losing a valuable business goal of offering integrated travel services as a part of the customer's portal environment.

Context

Distributed applications need to establish secure and trusted relationships between them to perform some work in a web-service environment which may be unreliable and/or insecure (e.g. the Internet). The concept of "Trusting A" mainly means "considering true the assertions made by A", which does not necessarily correspond to the intuitive idea of trust in its colloquial use.

WS-Security begins with the assumption that, if one of the parties uses a particular type of security token within the WS-Security header, then the other party will be able to interpret and process this token. A fundamental issue that WS-Security did not address is how two entities (a SOAP client and SOAP Service) can agree on the nature and characteristics of the security tokens that are the fundamentals of WS-Security.

Problem

Establishing security relationships is fundamental for the interoperation of distributed systems. Without applying relevant trust relationships expressed in the same way between the involved parties, web services have no means to assure security and interoperability in their integration. How can we define a way for the parties to trust each other's security credentials?

The possible solution is constrained by the following forces:

- ***Knowledge:*** In human relationships, we are concerned with first knowing a person before we trust her. That attitude applies also to web services. We need to have a structure that encapsulates some knowledge about the unit we intend to trust.

- ***Policy consideration*****:** The web service policy contains all the required assertions and conditions that should be met to use that web service. The trust structure should consider this policy for verification purposes.

- ***Confidentiality and Integrity*****:** Policies may include sensitive information. Malicious consumers may acquire sensitive information, fingerprint the service and infer service vulnerabilities. This implies that the policy itself should be protected.

- ***Message integrity***: The data to be transferred between the partners through messages may be private data that need to be protected. Attackers may try to modify or replace these messages.

- ***Time Validity***: For protection purposes, any interactions or means of communications (including the trust relationships) between the web services should have a time limit, that determines for how long the trust relationship is valid.

Solution

We define explicitly an artifact (security token) that implies trust. This artifact implies what kinds of assertions are required to make trustworthy interactions between the involved web services.

We should verify the claims and information sent by the requester in order to obtain the required security token that becomes a proof enough to establish a trust relationship with its target partners.

Structure

Figure 1 describes the structure of this pattern. **Claim** is a statement made about the attributes of a client, service or other resource (e.g. name, identity, key, group, privilege, capability, etc.). Claims are assertions, for example: "I am Joman", "I am an authenticated user and I am authorized to print in printer P". Claims are used to validate the requests made by a sender and need to be verified.

A **Security Token** is a collection of claims. It is possible to add signatures to tokens. **Security Token** also is a generalization of two types: **Signed Security Token** that is cryptographically endorsed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket) and **Proof-of-Possession (PoP)**

Token that contains a secret *data* parameter that can be used to prove authorized use of an associated security token and provides the function of adding digital signature. Usually, the proof-of-possession information is encrypted with a key known only to the recipient of the PoP token.

The **Security Token Service (STS)** is a web service that issues security tokens. It makes decisions based on evidence that it trusts. The **STS** is responsible for generating security tokens and, providing challenges for the requester to ensure message freshness (the message has not been replayed and is currently valid), verification of authorized use of a security token, and finally establishing, extending and removing trust in a domain of services. The **STS** is the heart of WS-Trust and forms the basis of trust brokering. The main output of the **STS** is a trust relationship between the requester and the receiver expressed as a security token. It represents the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes in a secure, reliable and time-relevant manner.

Each **STS** has a **Trust Engine** that evaluates the security-related aspects of a message using security mechanisms and includes policies to verify the requester's assertions. The **Trust Engine** is responsible for verifying security tokens and verifying claims against policies. A **Policy** is a collection of policy assertions that have their own name, references, and ID. Policies form the basic conditions to establish a trust relationship. Verifying the requester's claims against policy assertions generates an approval to use the target service. A policy may reference another policy (ies), in order to check the tokens sent by the requester or verified by the receiver.

Dynamics

We describe the dynamic aspects of the WS-Trust using sequence diagrams for the use cases "*create security token*" and "*access a resource using a token*".

Create a security token (Figure 2):

Summary: STS creates a security token using the claims provided by the requester.
Actors: A Requester
Precondition: The STS has the required policy to verify the requester claims and the requester provides parameters in form of *claims* and *RequestType* signed by a *signature*.
Description:

a. The requester requests a security token by sending the required *claims* and *RequestType* signed by a *Signature* to the STS. The signature verifies that the request is legitimate.
b. The STS contacts the Trust Engine to check the requester's claims.
c. The Trust Engine contacts the web service's policy to verify the claims including attributes and security token issuers of the requester.
d. Once approved, the STS creates a security token containing the requested claims.
e. The STS sends back its *SecurityTokenResponse* with a security token issued for the requester.

Postcondition: The requester has a security token that can be used to access resources in a trusted unit.

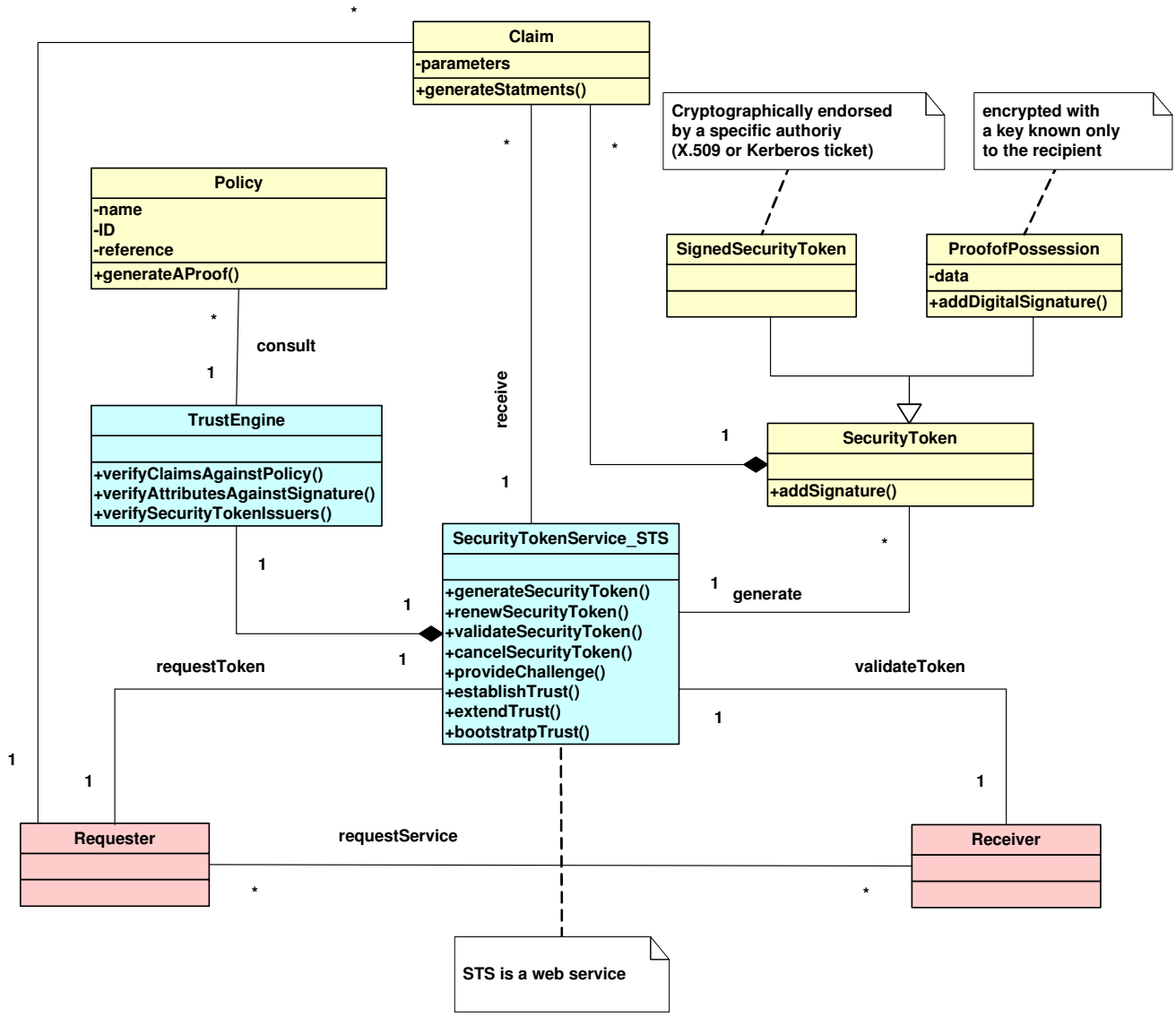

Figure 1: Class Diagram for the WS-Trust Pattern

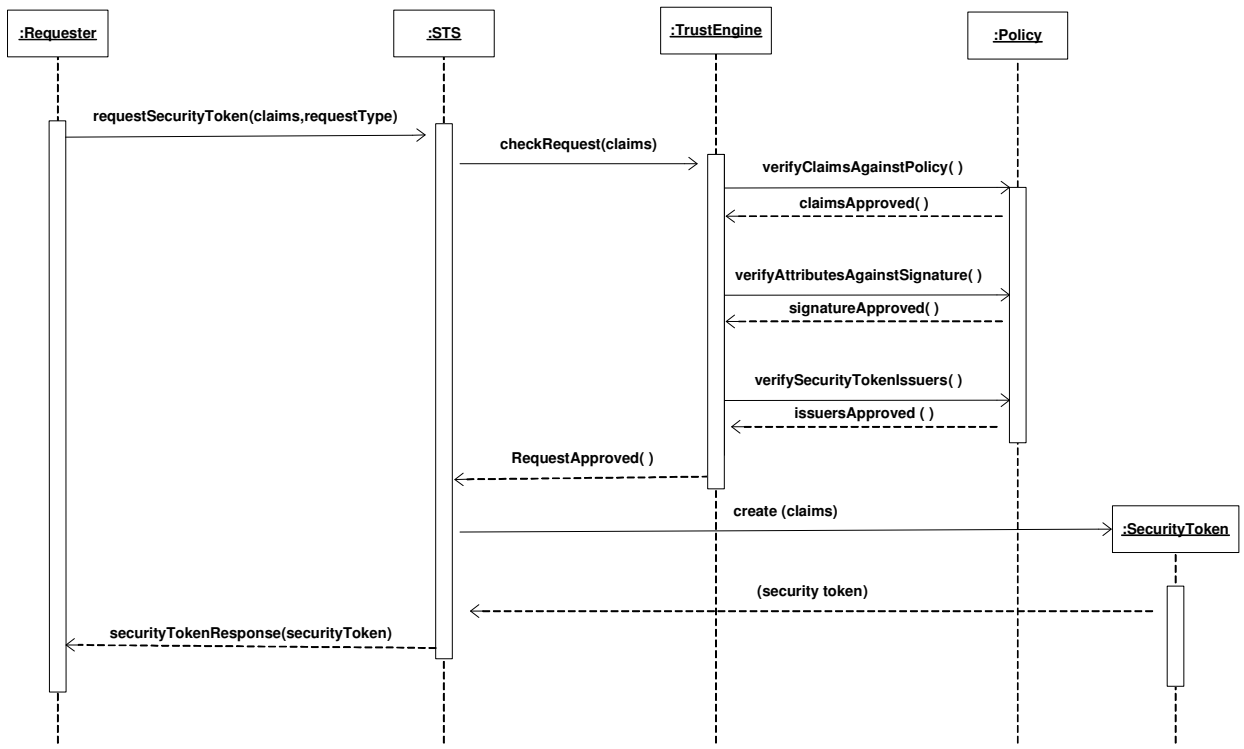

Figure 2: Sequence Diagram creating a security token

Access a resource using a token (Figure 3):

Summary: A STS allows the use of resources by establishing trust by verifying *proofOfClaims* sent by the requester.
Actors: A Requester
Precondition: The Trust Engine has the required policy to verify the requester' security token.
Description:

a. The requester asks for a service access by providing the required security token.
b. The receiver sends the security token to the STS for verification.
c. The STS use its Trust engine to verify the security token claims.
d. Once approved, the STS notifies the receiver that the security token is valid and verified.
e. The receiver gives the requester a token that implies the right to use the service.

Postcondition: The requester has a security token that can be used to access services in a Receiver web service.

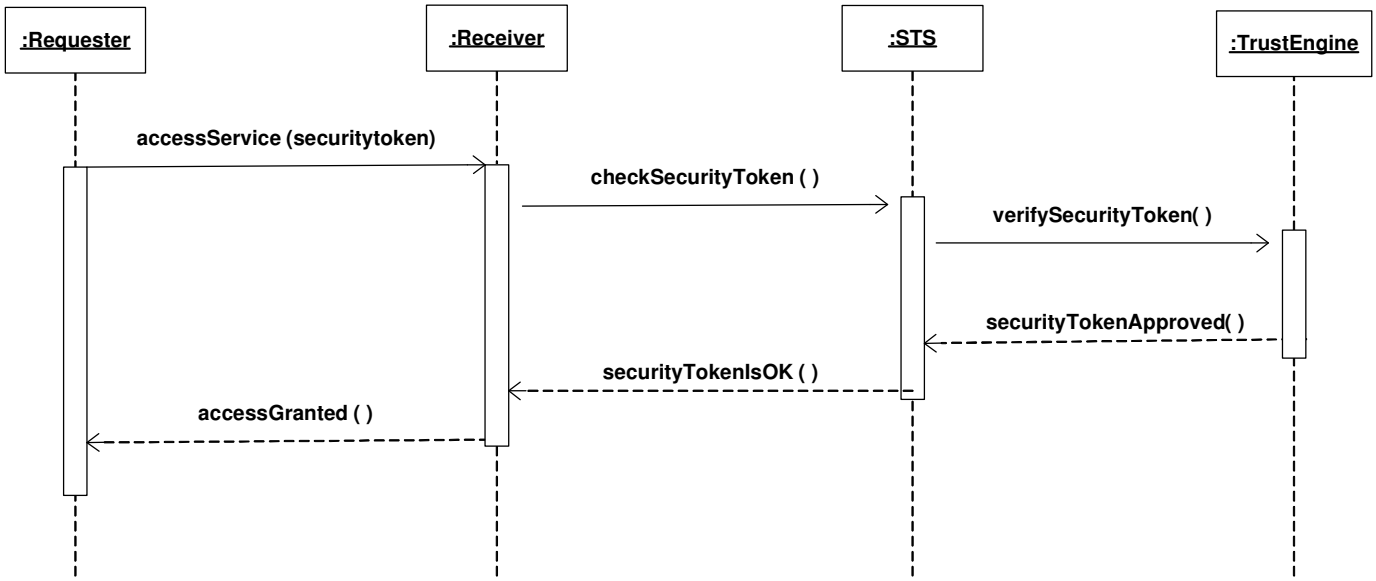

Figure 3: Sequence Diagram accessing a resource using a token

Implementation

In this solution, the concept of trust is realized by obtaining a security token from the web service (in our diagram, the Security Token Service) and submitting it to the receiver who in turn validates that security token through the same web service. Upon approval, the receiver establishes a valid trust relationship with the receiver that lasts as long as the security token is valid.

In order to assure effective implementation, we need to take in consideration the following:

- To communicate trust, a service requires proof, such as a signature to prove knowledge of a security token or set of security tokens. A service itself can generate tokens or it can rely on a separate STS to issue a security token with its own trust statement.

- Although the messages exchanged between the involved entities are protected by WS-Security; still three issues related to security tokens are possible: security token format incompatibility, security token trust, and namespace differences. The WS-Trust pattern addresses these issues by defining a request/response protocol (in which the client sends *RequestSecurityToken* and receives *RequestSecurityTokenResponse)* and introducing a Security Token Service (STS) which is another web service.

- Based on the credential provided by the requester, there are different aspects of requesting a security token (RST), each of which has a unique format that the requester should follow:

- The issuance process: formed as *RequestSecurityToken (RequestType, Claims).*This is our use case Create a security token in the Dynamics section.
- The renewal process: formed as *RequestSecurityToken (RequestType, RenewTarget).*
- The cancel process: formed *RequestSecurityToken (RequestType, CancelTarget).*
  By the way, the cancelled token is no longer valid for authentication and authorization.
- The validate process: formed as *RequestSecurityToken (RequestType, ValidateTarget).*

.

The WS-Trust specification was created as part of the Global XML Web Services Architecture (GXA) framework, which is a protocol framework designed to provide a consistent model for building infrastructure-level protocols for web services and applications [Box02]. It was authored by Microsoft, IBM, Verisign, and RSA Security and was approved by OASIS as a standard in March 2007.

Example Resolved

Ajiad now has the ability to automate its trust relationships with its partners by managing the registration tasks for all its partners and issuing customers a unique ID's. In this case, *Ajiad* provides a mediator between the customers and its participant partners and plays the role of negotiator and third-party player who is trying to satisfy both sides.

Ajiad now can offer a Security Token Service for its business partners, who may find useful ways to take advantage of credit processing and other services offered by *Ajiad*, which now has new business opportunities.

Consequences

The WS-Trust pattern presents the following *advantages:*

- ***Security.*** By extending the WS-Security mechanisms, we can handle security issues such as security tokens (the possibility of a token substitution attack), and signing (where all private elements should be included in the scope of the signature and where this signature must include a timestamp).

- ***Trust.*** With this solution, we have the choice of implementing the WS-Policy framework to support trust partners by expressing and exchanging their statements of trust. The description of this expected behavior within the security space can also be expressed as a trust policy.

- ***Confidentiality.*** We can achieve confidentiality of users' information. Since Policy providers now can use mechanisms provided by other web services specifications such as WS-Security [ibm09b] to secure access to the policy, XML Digital Signature [w3c08] to authenticate sensitive information, and WS-Metadata Exchange [w3c09].

- All the security tokens exchanged between the involved parties are signed and stamped with unique keys that are known only to the recipients.

- ***Time validity***. We can specify time constraints in the parameters of a security token issued by STS. This constraint will specify for how long that security token is valid. Upon expiring, the security token's holder may renew or cancel it.

The WS-Trust pattern presents the following *liabilities:*

- The efficiency of WS-Trust may suffer from the repeated round-trips for multiple token requests. We need to make an effort to reduce the number of messages exchanged.

- The WS-Trust standard is a lengthy document and several details were left to avoid making the pattern too complex. The interested reader can find more details in the WS-Trust Standard web page [oas09].

Known Uses

- DataPower's XS40 XML Security Gateway [dat05] is a device for securing web services that provides web services access control, message filtering and field-level encryption. It centralizes policy enforcement, supporting standards such as WS-Security, WS-Trust, WS-Policy and XACML.
- SecureSpan™ XML Firewall [lay09] enforces WS* and WS-I standards to centralize security and access requirements in policies that can be run as a shared service in front of applications.
- Vordel Security Token Service [vor09] is used to issue security tokens and to convert security tokens from one format to another. The security tokens created by an STS are bound to the messages travelling between web services..
- PingTrust, a standalone WS-Trust Security Token Server [pin06] creates and validates security tokens that are bound into SOAP messages according to the Web Services Security (WSS) standard.

Related Patterns

- The *Trust* Analysis Pattern, [Fay04]. The objective of this pattern is to provide a conceptual model that embodies the abstract aspects of trust to make it applicable to different domains and applications.

- The *Credential* Pattern [Mor06]. This pattern addresses the problem of exchanging data between trust boundaries and how to resolve the problem of authenticating and authorizing a principal's identity over different systems.

- The *Circle of Trust* pattern allows the formation of trust relationships among service providers in order for their subjects to access an integrated and more secure environment [Del07]. The WS-Trust pattern could be used to establish trust between providers.

3. Conclusions

This pattern describes how to build trust relationshsips and how existing trust relationships may be used as the basis for brokering trust through the creation of security token issuance services. These security token issuance services build on WS-Security to transfer the requisite security tokens in a manner that ensures their integrity and confidentiality.

Future work will include designing patterns for other web services standards such as WS-Federation and WS-SecureConversation that depend on WS-Trust as a prerequisite foundation. This will give us a good chance to analyze and discover how WS-Trust fits with other web services standards and how much it could simplify the implementation of theses specifications in real-life business applications.

Acknowledgements

We thank our shepherd Takao Okubo for his comments. We also thank our PC member Yann-Gael Gueheneuc for his discussion about the value of patterns describing standards.

References

[Box02] D. Box, *Understanding GXA,* Microsoft Corporation, http://msdn.microsoft.com/en-us/library/aa479664.aspx - Last accessed on December 15, 2009

[dat05] IBM Corporation, WebSphere DataPower XML Security Gateway XS40, http://www-01.ibm.com/software/integration/datapower/xs40/ – Last accessed at November 25, 2009

[Del07] N. Delessy, E.B.Fernandez, and M.M. Larrondo-Petrie, "A pattern language for identity management", *Procs. of the 2nd IEEE Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007),* March 4-9, Guadeloupe, French Caribbean.

[Fay04] M.E.Fayad, and H. Hamza, "The Trust Analysis Pattern*", in Proceedings of the Fourth Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP 2004),* Porto Das Dunas, Ceara, Brazil. August 10-13, 2004. http://sugarloafplop2004.ufc.br/acceptedPapers/ww/WW_1.pdf - Last accessed on December 15, 2009

[Fer06] E.B.Fernandez and N. Delessy, ""Using patterns to understand and compare web services security products and standards", *Proceedings of the Int. Conference on Web Applications and Services (ICIW'06*), Guadeloupe, February 2006. IEEE Comp. Society, 2006.

[ibm09a] Security in a Web Services World: A Proposed Architecture and Roadmap, http://download.boulder.ibm.com/ibmdl/pub/software/dw/library/ws-secmap.pdf - Last accessed on December 3, 2009

[ibm09b] IBM Corporation, Web Services Security 2004, http://www.ibm.com/developerworks/library/specification/ws-secure/ – Last accessed on December 07, 2009

[lay09] Layer 7 Technologies, The SecureSpan XML Firewall, http://www.layer7tech.com/main/products/xml-firewall.html – Last accessed on December 09, 2009

[Mad03] WS-Trust: Interoperable Security for Web Services, by Paul Madsen, http://www.xml.com/pub/a/ws/2003/06/24/ws-trust.html - Last accessed on November 30, 2009

[Mor06] P. Morrison and E. B. Fernandez, "The credentials pattern*", in Proceedings of the 2006 conference on Pattern languages of programs (PLoP 2006),* Portland, OR, USA. October 21–23, 2006. http://portal.acm.org/citation.cfm?id=1415472.1415483 - Last accessed on December 15, 2009

[oas06] OASIS, Web Services Security: (WS-Security 2004), http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf - Last accessed on December 15, 2009

[oas09] OASIS Standard, WS-Trust 1.4, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.pdf - Last accessed on December 07, 2009

[pin06] Ping Identity Corporation, PingTrust, a standalone Security Token Server, http://www.pingidentity.com/about-us/news-press.cfm?customel_datapageid_1173=1404 - Last accessed on December 15, 2009

[vor09] Vordel Limited, Vordel STS, http://www.vordel.com/solutions/security_token_services.html - Last accessed on December 15, 2009

[w3c07] W3C, Web Services Policy 1.5 – Framework, 4 September 2007, http://www.w3.org/TR/ws-policy/- Last accessed on December 15, 2009

[w3c08] W3C Working Group, XML Signature Syntax and Processing (Second Edition) 2008, http://www.w3.org/TR/ws-gloss/ – Last accessed on December 15, 2009

[w3c09] W3C Working Draft 2009, Web Services Metadata Exchange, http://www.w3.org/TR/ws-gloss/ – Last accessed on December 15, 2009