Identity Bridging Techniques across SOA-based Business Service Networks

Mamoon Yunus and Rizwan Mallal
Crosscheck Networks
Newton, Massachusetts
U.S.A
{mamoon, rizwan}@crosschecknet.com
http://www.crosschecknet.com

Abstract

Identity Management is a critical aspect of deploying secure SOA-based Business Services Networks. Establishing trusted Business Services Networks require application- and user-level authentication and authorization of invoked services. In effective BSNs, service invocations should seamlessly traverse corporate boundaries. With loosely coupled and chained Web Services, building trusted Business Networks require flexibility in Identity Management across protocols and messages. As corporate boundaries become porous to trading partner interactions, identity enforcement and identity bridging become central in ensuring Business Service Network flexibility without compromising trust-based security.

Keywords: Identity Management, Message-based Identities, Protocol-based Identities, Identity Bridging, Web Services, SOA.

1. Introduction

Successful enterprises change rapidly and establish efficient, loosely coupled Business Services Networks with a large community of trading partners in their supply and demand chains. Such integration increases the demand for sophisticated, nimble and well-integrated applications, architectures and standards-based platforms. Reducing cost and increasing revenue – fundamental to successful enterprises – require agile and rapid connections between internal business systems and external trading partners. Enterprise IT Architects enable such nimble business interactions through standards-based, interoperable and reusable application interfaces. This standards-based, reusable model is the basis of a Service Oriented Architecture (SOA).

The principles of the SOA are not new. The goal to move away from a monolithic software architecture to a more modular and interoperable software architecture has always been one of the key principles of SOA. SOA paradigm was applied to DCOM, DCE and CORBA specifications in the 1990s. However, the SOA paradigm never established a solid foothold because of interoperability issues within and across such specifications.

Today, the SOA paradigm is again being applied to developing Business Services Networks. However, this time SOA is gaining foothold in the enterprise market because of Web Services. Web Services-based SOAs are enabling businesses to build and streamline their applications through a standard set of technologies and protocols [1].

Web Services describe a standard way of integrating applications using SOAP, WSDL and UDDI open standards over any protocol. The Web Services Stack is a collection of standards that is used to define, locate and implement messaging between applications. A Web Services Stack is comprised of four areas: service transport, messaging, service description, and services discovery. Web Services are independent of the communication layer and may use communication protocols, such as HTTP, HTTPS, JMS, SMTP, FTP, and IIOP for transporting messages between networked applications.

Web Services provide unprecedented standards-based flexibility for loosely coupling systems to establish a BSN. The flexibility is available both for transport protocol as well as message representation using XML and SOAP. This protocol and message level flexibility in turn requires security provisions to ensure that information privacy, integrity, and threat mitigation [2] are established within the BSN. Identity Management across business domains forms the basis of a secure SOA-based BSN. Most businesses and individuals engage in business transactions and activities by first establishing mutual identity. Therefore, the need for flexible identity bridging across multiple identity domains becomes a fundamental issue is establishing integration with trading partners within a BSN.

2. Evolution of Identity Management Systems

Identity Management for User-to-Application interaction is a well-known domain that has been successfully addressed in the industry through

standards such as LDAP [3]. The LDAP protocol provides a clear representation of users, group and access control lists and the ability for applications to invoke and consume such entities. LDAP servers such as OpenLDAP [4], SunOne Directory Server, Oracle OID are commercial grade LDAP servers with tight integrations to Application Servers and Web Servers. The rapid proliferation of such identity stores within large enterprises fractured identity management, with multiple identities required for internal users across numerous application silos.

The identity fragmentation caused a further identity crisis with the emergence of web-based customer interaction that required information access to multiple internal CRM, ERP, SCM and RDBMS systems all maintaining their own identity information. The multiple identity crises were eventually addressed with the emergence of Single Sign On (SSO) vendor products such as Netegrity (now CA) SiteMinder™, Oblix (now Oracle) CoreId™, IBM Tivoli Access Manager™, and RSA ClearTrust™. SSO systems provide a unifying interface for multiple identity stores and remove the need for internal and external users to maintain multiple identities while interacting with a variety of business applications.

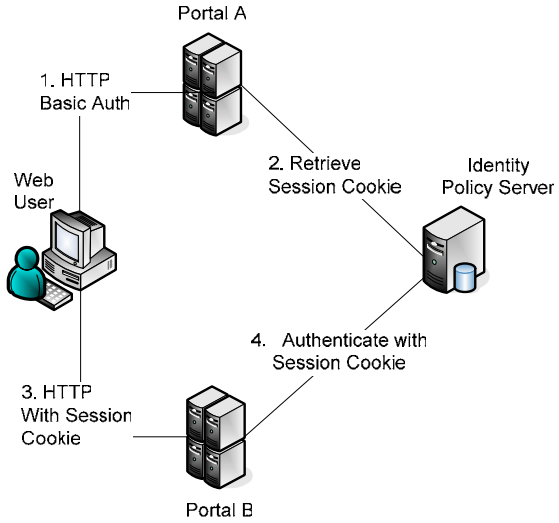

Figure 1: Typical Single Sign On (SSO) Process.

Single Sign On systems such as SiteMinder™ require users to provide authentication credentials once. Figure 1 displays a typical Single Sign On process. In step 1, a web user signs into Portal A by providing credentials such as username and password. As shown in step 2, the credentials are passed to the Identity Policy Server that generates a session cookie or an artefact. The Policy Server also provides authorization decision to ensure that the authenticated user is indeed allowed to access the desired URI. From then on, the artifact serves as the credential token. The artifact is passed to other websites, such as Portal B in step 3. The user is no longer required to sign in to other Portal as long as the session token or the artifact is valid and has not expired [5].

Single Sign On Systems are widely deployed across enterprises. Such systems have developed comprehensive functionality in maintaining roles, and protecting resources within a User-to-Application (U-to-A) domain, however, they lack functionality in Web Services-based Application-to-Application (A-to-A) identity management.

3. Web Services Identity Crisis

With the evolution of SOA to standards-based Web Services technologies, the need for flexible Identity Management has become clear. In the U-to-A mode, identities typically traverse networks over protocols: HTTP Basic Authentication, HTTPS Mutual Authentication, or more sophisticated 2-Factor Authentication is used over TCP/IP.

However, with Application-to-Application (A-to-A) interaction, the identity management dimensions increase across protocols and messages. A-to-A interactions occur over diverse protocols such as HTTP, HTTPS, SMTP, FTP, JMS and IIOP with protocol-based identities exchanged between applications. In addition to protocol-based identities, applications exchange Web Services-based identity tokens using standards such as WS-Security 1.1 with Username, X.509, Kerberos and SAML profiles. Such identities tokens are embedded in the message are independent of the protocol.

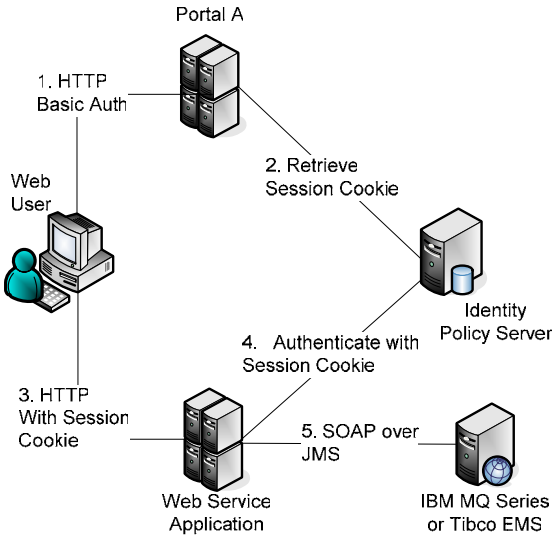

Figure 2: Single Sign On Process with Web Services SOAP over JMS.

Figure 2 illustrates a typical system that involves a user authenticating to a Portal A severed by a web

application such as BEA WebLogic. An identity agent within the WebLogic Application Server presents the user credentials to the Identity Management Policy Server. Based on the credentials, the Identity Policy Server authenticates and authorizes access to the URI. Then, a session cookie is generated by the Policy Server and is passed to the user, as shown in step 2. The subsequent invocation, step 3, is a Web Services call over HTTPS containing the session cookie. The Web Service endpoint URI is also protected by the identity agent of the Web Services Application that takes the session cookie and calls the Policy Server to check for the authenticity and validity of the session cookie, as shown in step 4. If the cookie has expired, or is invalid, access is denied to the resource. Similar to Figure 1, the Web Service is a URI and may be subjected to access control. As shown in step 5, the Web Services in turn may send the SOAP request to a back-end legacy mainframe using JMS-based messaging without security provisions for privacy, integrity and access control.

Current identity systems provide Web Services support for authentication, authorization and access control of Web Service endpoints. However, they fall short in providing WSDL-awareness and easy-to-deploy identity bridging across protocols and messages. The following sections delineate techniques to extend existing Identity Systems to meet the needs of SOA-based BSNs.

4. Web Services Aware Identity Systems

Identity Management systems have their roots in SSO and enabling easy user access to multiple websites and portals without the need to log in multiple times. The SSO capability focuses on locking down URI resources and ensuring that the user has the right privileges to invoke the URI. Identity Management Policy Servers are not Web Services aware. They lack facilities to load WSDLs and secure access privileges based on operation names. Identity systems can provide access restrictions on an endpoint, however, Web Services deployed demand more granular control at the operation level.

To attain operation-level granular control, existing identity management systems can be extended by appending URI resources in the Policy Server to include a name-value pair for the invoked operation. When a Web Services client invokes an operation, e.g., *getPrice*, the Web Service Application's identity agent sends an extended URI to the Policy Server: *http://application?operation=getPrice*. The Policy Server, with the manually extended URIs checks for a match against the URI assembled by the Web Service Application. Such techniques serve as a stop-gap measure until Policy Servers can parse WSDLs and enable SOA Security personnel to restrict access by associating operation-level resources to roles and identities without the need to manually load extended URIs with appended operation names – an arduous task, especially for complex WSDLs with hundreds of operations names.

5. Identity Representations

Identity representation in the protocol domain primarily constitutes of HTTP Basic Authentication, HTTPS Mutual Authentication, HTTP(S)-based session cookies, and JMS Attributes populated with credentials. Message based credentials include WS-Username tokens, WS-X.509, WS-Kerberos, WS-SAML assertions [6]. Non-standards-based credentials may be presented within a message at any arbitrary location. Such credentials can be selected from the message through XPath [7].

6. Identity Bridging Techniques

The need for Web Services Gateways that handle complex identities representations, tie in to existing Identity Management Systems, and easily bridge between different identity domains is apparent for building SOA-based BSNs. The complexities are compounded by protocol mixing and message-level representation of identities with the need to seamlessly traverse multiple security boundaries for a single transaction.

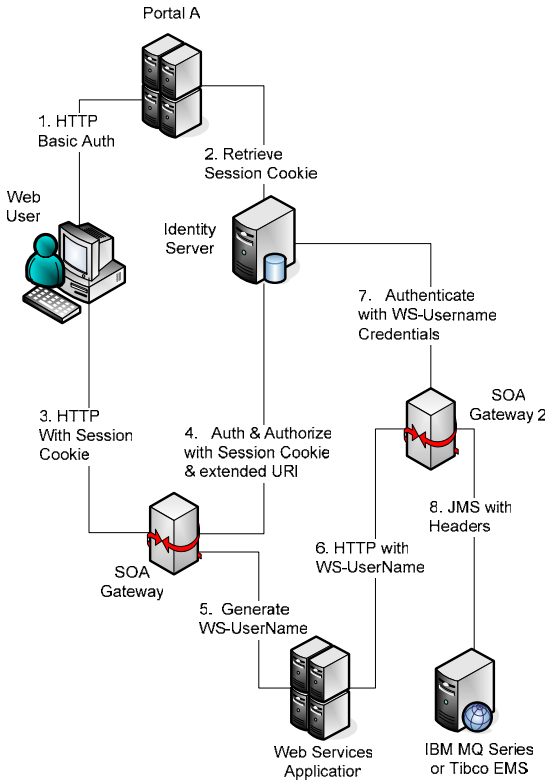

Figure 3: Single Sign On Process with SOA Gateway performing Identity Bridging.

SOA Gateways play a crucial role in removing the burden of identity bridging from application developers who can remain more focused on business functionality.

Figure 3 depicts and extension of Figure 2 with SOA Gateways introduced to handle protocol mixing and identity bridging for end-to-end Identity management. The Gateway policies alleviate the authentication and authorization burden from the Web Service Application. An Identity bridging tasks such as moving from session token-based authentication to WS-Username token-based authentication is shown in steps 3, 4, and 5. Sophisticated SOA Gateways are Web Services aware and can digest Web Services-based identity tokens, step 6, and in turn present them to the Policy Store to further authentication and authorization decisions, step 7. SOA gateways also bridge between message-level identities to protocol-level identities as shown in step 8, where the SOA Gateway 2 converts WS-Username tokens into JMS Headers. In the following paragraphs, a number of popular Identity bridging techniques are described in detail.

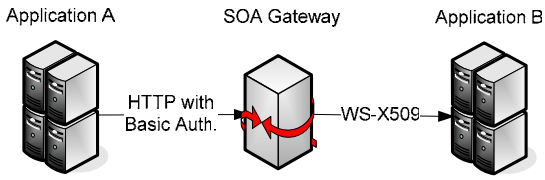

Figure 4: Protocol-based HTTP Basic Authentication to message-based WS-X.509 identity bridging.

Technique #1: *HTTP Basic Authentication→WS-X.509 Identity* bridging technique shown in Figure 4, bridges protocol-based HTTP Basic Authentication credentials from invoking Application A to WS-X.509 message-based credential required by Application B. The SOA Gateway acts as the Identity Bridge between Application A and Application B. The X.509 certificate selected by the SOA Gateway for insertion into the SOAP Header may be a static Gateway X.509 certificate such that the SOA Gateway now vouches on behalf of Application A. Such bridging modes reduce the authentication burden on the target Application B. For more granular identity control, the SOA Gateway can pick an X.509 from a key-pair associated with the credentials presented by Application A. In both scenarios, Application B is responsible for digesting the WS-X.509 message header and checking for the validity and integrity of the X.509 certificate presented through certificate chain traversal as well as CRL checking [8].

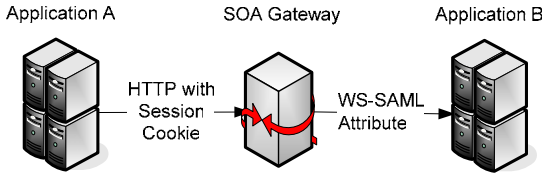

Figure 5: Protocol-based HTTP Session Cookie to message-based WS-SAML identity bridging.

Technique #2: *HTTP(S) Session cookies – SAML Attribute* bridging technique shown in Figure 5. In this technique, Application A first obtains a session cookie through a SSO and shown in Step 2 of Figure 3. Once Application A has the session cookie, it is presented to the SOA Gateway that checks for validity of the session cookie by calling the SSO Policy Server. The SOA Gateway then takes the SOAP request and inserts a SAML assertion in the header with the session cookie injected within a SAML attribute. Additional entitlement attributes may also be injected as additional SAML attributes. Application B then digests the SAML from the SOAP header and can retrieve the session cookie for further authentication. Application B may receive the WS-SAML message independent of protocol. HTTP(S) and JMS-based asynchronous protocols are most widely used. For additional integrity control, SOA Gateway may sign the SAML assertion in the SOAP Header. For details on signing SAML assertions, see [5].

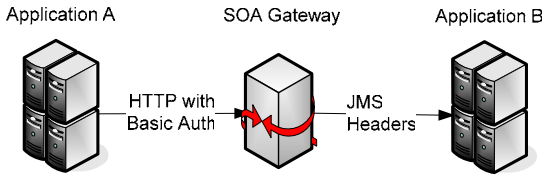

Figure 6: Protocol-based HTTP Basic Authentication to protocol-based JMS-Header identity bridging.

Technique #3: *HTTP(S) Basic Auth – JMS Header* bridging technique is shown in Figure 6. In this technique, Application A present HTTP Basic Authentication credentials to the SOA Gateway that authenticates Application A, takes its credentials such as username, and injects it as a JMS Header [9]. Using this technique enables protocols such HTTP and JMS to bridge identity information seamlessly. This technique is extended by taking

element from SOAP requests, such as SAML attributes and mapping them into JMS Headers.

In large enterprises with significant legacy IT assets, JMS-based messaging is prevalent through commercial products such as IBM MQ Series™, Tibco EMS™ and SonicMQ™. Moving across different protocol-based identities and message-based identities is a requirement for establishing SOA-based BSNs.

7. Future of WS-Identity Management

Commercial SSO products are gradually evolving towards Web Services awareness. The most significant near-term focus is around enabling Policy Servers to load and manage WSDLs for granular operation-level control. This WSDL awareness will enable centralized administration of Web Services Identities within a corporate SOA. New standards, such as WS-Trust promise extending SOA Identity Management into BSNs. Through WS-Trust, a SOA gateway can query Policy Servers anywhere in a BSN beyond the local enterprise Policy Server. WS-Trust defines a request-response protocol that enables SOA gateways and Policy Enforcement Points (PEPs) to resolve security token format incompatibility and establish security token trust [10]. Companies such as Ping Identity offer a robust WS-Trust Token Servers, PingTrust™ that simplifies the integration burden into identity stores within BSNs.

WS-Policy is another emerging standard that provides a way to express required characteristics of a Web Services endpoint. For example, for identity management WS-Policy provides the provisions to express what security tokens is the Web Services capable of processing, and what security token does the Web Service prefer [11].

Another area of standards development, WS-SX [12] extends the security token profiles already defined in WS-Security to include policy representations for multi-domain trust brokering by tying message security representations with policy (WS-Policy) and trust representations WS-Trust.

8. Conclusion

Building successful Business Service Networks requires use of ubiquitous standards and ease of traversal across boundaries and domains within a corporation and across business entities. Identity Management is critical is establishing secure BSNs. However, with identities representation fragmented across enterprises, identity bridging becomes a central focus of establishing business links. Using SOA gateways provide a cost effective, rapid and secure way of identity bridging. Alternative techniques are programming intensive and result in tying security logic within business logic, a practise that is difficult to maintain and may result is security breaches. It is expected that in the near future, Single Sign On focused Policy Server will become more Web Services friendly decreasing the burden on the SOA gateways and other PEPs to provide WSDL operation-level access control.

Significant development and activity within the standards community to normalize token policy representations (WS-Policy), trust brokering (WS-Trust) and secure messaging (WS-Security) through WS-SX shows that identity bridging is crucial in realizing the benefits of a SOA-based Business Service Network.

9. References

[1] Ed Ort, Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools. *Sun Developer Network*, April 2005.

[2] Mamoon Yunus, Rizwan Mallal, "An Empirical Study of Security Threats and Countermeasures in Web Services-Based Services Oriented Architectures," *Lecture Notes in Computer Science*, Volume 3806, Oct 2005, Pages 653 – 659.

[3] Brian Arkills, LDAP Directories Explained: An Introduction and Analysis, Addison-Wesley, ISBN 0-201-78792-X, Pages 3-37.

[4] Kurt Zeilenga, Richard Krukar, The OpenLDAP Project, http://www.openldap.org/.

[5] John Hughes, Eve Maler, et. al., Sercurity Assertion Markup Language (SAML) V2.0 Technical Overview, Working Draft, http://www.oasis-open.org/committees/download.php/14361/sstc-saml-tech-overview-2.0-draft-08.pdf, September 2005, Section 2.

[6] Kevin Lawrence, Chris Kaler, et. al., Web Service Security 1.1, OASIS Public Review Draft, http://docs.oasis-open.org/wss/v1.1/, June 2005.

[7] James Clark, Steve DeRose, XML Path Language Version 1.0, W3C Recommendation, http://www.w3.org/TR/xpath, October 1999.

[8] Blake Douranee, XML Security, McGraw Hill/Osborne, ISBN 0-07-219399-9, Pages 46-55.

[9] Mark Hapner, Rich Burridge, et.al., Java Message Service Version 1.1, Sun Microsystems, April 2002, Pages 30-35

[10] Steve Anderson, Jeff Bohren, et. al., Web Services Trust Language (WS-Trust), ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, February 2005.

[11] Sidharat Bajaj, Don Box, et. al., Web Services Policy Frameworks (WS-Policy), http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf, March 2005.

[12] Kevin Lawrence, Chris Kaler, et. al, OASIS Web Services Secure Exchange (WS-SX) TC, http://www.oasis-open.org/committees/ws-sx/