Developing Secure Web Services for Computational Portals

Choonhan Youn, Marlon Pierce and Geoffrey Fox
Community Grid Labs, Indiana University
cyoun@ecs.syr.edu, {marpierce, gcf}@indiana.edu
Postal Address: 501 N. Morton Street
Bloomington, IN 47401
Phone for Corresponding Author: (812) 856-0755
FAX: (812) 856-1537

Abstract

Computational web portals provide uniform access to remote computational resources--hardware, software, and data--by hiding the complexity of the heterogeneous, distributed, high performance computing back end. These portal services may be implemented in a programming-language and platform independent way using a Web services approach. However, security in Web services for distributed computing systems is an open issue involving multiple security mechanisms and competing standards. In this paper we present an implementation of a flexible, message-based security system that can be bound to multiple mechanism and multiple message formats.

Keywords: Security, Web service, SAML, Kerberos, GSS-API

1. Introduction

Scientists and researchers can interact with the computational Grids such as Globus [1] via portal mechanisms [2]. For accessing those remote resources, one can view the Web Services approach [3, 4] as a "kit" of services, which can be used in different ways in different applications. Seamless access in a computing portal is an important service, which is aimed at allowing applications to be run on arbitrary appropriate backend resources. Such seamless capability is rather natural in a computing web-based architecture.

Basically, a computing portal [5] is an application that integrates access to the data, computers and tools needed for a particular computational science area. It comprises an infrastructure enabling scientists to use a diverse set of distributed services that access remote compute resources. We may identify the general services needed for the computing portal. First, a security service allows the authenticated user to use the distributed remote resources through the portal. Second, a discovery service such as object lookup and registration allows a user to find the usable and available service in the portal. Third, an information service such as object persistence and database support allows a user to store and recover the portal data. Fourth, a job tracking service such as event and transaction allows a user to monitor and track the job status from the batch scheduler. Fifth, a file service provides a user for managing the user directory such as copy, rename, move, and so on. Sixth, a job composition service such as application integration allows a user to compose the core services. Finally, there are collaboration services [6]. These service capabilities in the portal support the scientific problem solving process for users. For example, using the Application Web Services [7], legacy applications can be easily wrapped into the portal services that are available to appropriate users, including other portals. From the user's point of view, it is very useful for the user to adapt the scientific applications into the portal. We have described how we design and implement core Web Services and Application Web Services based on the service-oriented architecture for processing the complex scientific problem solving [8].

Security in such an environment is critical, since it often involves directly accessing a user's resources on a particular computer through delegation to a middle tier proxy. In this paper, we describe how message level security may be incorporated into the architecture of a Web service based computing portal. We must address two important problems. First, how do we add security to Web service messages, over and above transport level security? Second, how can we provide an adaptable system that works with different security message formats and mechanisms? We need secure SOAP messages between user interface server and the repository and the service provider for the user authentication, based on the message-level security architecture. SOAP security should be provided through standard interfaces to specific mechanisms such as Kerberos [9], Shibboleth [10], PKI [11], or Globus GSI [12]. The general approach is to use the assertion based security such as SAML, WS-security into SOAP messages. An assertion, for example, SAML, WS-Security, is an XML document describing the information about authentication acts performed by subjects, attributes of subjects and authorization decisions, created with a specific mechanism. To this, we must add additional metadata that can be used to forward the secure assertion to the correct handler classes.

2. Web services for computing portals

Web Services support the commercial and industrial companies which have been the long sought goal for the standard model for providing the way of e-business and e-science computing services [3]. Web services are also important to the Grid computing community, forming the basis for the Open Grid Services Architecture (OGSA) proposed by the Global Grid Forum [13].

Most computing portals, such as older versions of our Gateway portal system [14] are based on a three-tired architecture and so have a classic stove-pipe problem in aspects of services. In order to integrate distributed services, the computing services should be designed for the interoperability and reusability. For addressing these challenges, we have presented a Web Service based computing portal architecture around Web Services model which have emerged as a popular standards-based, and service-oriented framework for accessing network-enabled applications [8, 15]. Web Services has an XML-based, distributed computing paradigm to address the heterogeneous distributed computing services. It defines a technique for describing a service component, accessing a service, and discovering a service. Its standards are being defined within the W3C. Web Services constitute three pieces with following standards: SOAP, WSDL, and WSIL (or UDDI).

- The Web Services Description Language (WSDL) [16] is a XML document for describing a Web Service, that is, service interface definition language.
- The Simple Object Access Protocol (SOAP) [17] is an XML-messaging, transport-independent protocol, and one means of formatting a Web Service invocation as a RPC mechanism.
- The WS-Inspection Language (WSIL) [18] is used for a distributed Web Service discovery, while the Universal Description, Discovery and Integration (UDDI) [19] specification is used for a centralized Web Service discovery. Both are served as the service discovery for a Web Service.

3. Secure Web services

Computational Web Portals may be built out of Web services such as job submission, context management, file service and application Web services [8]. Access to remote resources is performed through portal services that consist of Web services acting as the middle ware. These Web services pieces make no provisions for authentication, message integrity, authorization service (access control), and other security concerns. GSI [12] is based on the public key infrastructure which is used for X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. GSI SOAP [20] is available, which provides a GSI enabled HTTP protocol, GSI delegation, and authentication capabilities.

SOAP-based Web services do not directly provide message level security but do provide an extensibility mechanism: arbitrary additional XML messages such as security assertions may be included in SOAP headers. We are investigating a general purpose way of securing SOAP messages that support multiple underlying mechanisms such as Kerberos, PKI, and GSI independently based on user assertions using SAML, the Secure Assertion Markup Language [21].

We have developed a prototype system to support a secure Web services through the single sign-on (SSO). SSO allows the use of a secure resource at the destination site without directly authenticating to it: SSO requires secure delegation of initial authentication credentials. As the example of our prototype system, we have used Kerberos as a security mechanism and SAML as an assertion, but we have attempted to keep our design general for the adaptive control and will add support for other mechanism such as PKI and Globus GSI.

3.1. Web services security languages

SAML, an OASIS standard, is an XML-based security services framework for exchanging authentication and authorization information. Assertions are mechanism-independent, digitally signed information about authentication acts performed by subjects, attributes of subjects, and authorization decisions about whether subjects are allowed to access certain resources.

WS-Security [22] is a Web services security language as the basis for the construction of a wide variety of security models including PKI, Kerberos, and SSL and encryption technologies. It enables applications to construct secure SOAP message exchanges and describes a single-message security language that provides for message security that may assume an established session, security context and/or policy agreement.

Our authentication system is based on SAML as Web services security language. Although not included in the prototype, we have designed it to support other Web service security language such as WS-Security. A SAML-consuming service can accept the signed assertion or use it to find locations of Kerberos proxy tickets or grid proxy certificates. SAML can also be used to convey access control decisions made by other mechanisms, such as Akenti [23, 24]. SAML assertions are added to SOAP messages. We have implemented the SAML specification schema in Java using Castor [25], which can be used to convert between XML schemas and Java data objects. We have also developed client and server

applications for handling SOAP with SAML messages.

3.2. Secure SOAP message format

We are implementing a message-level security system using the Web services security language such as SAML and WS-Security, and the mechanisms such as Kerberos, PKI, Globus GSI. In order to support these different formats and mechanisms, we have added additional metadata to the SOAP header: we configure the message format which can identify the assertion and security mechanism.

As the language identifier, we are using the name tag, "Saml" for SAML and "WSSecurity" for WS-Security. This is followed by a tag "SignedAssertion" that contains the signed and perhaps encrypted security assertion. According to this name tag, SOAP server can extract and parse the user information and authorization for resources. In order to encrypt and decrypt the signed SOAP message (signed assertion and SOAP body message), we also need to specify what kind of the security mechanism is used to get a shared key session object. We thus provide an additional tag, "SecurityMechanism", "Kerberos" for Kerberos, and "PKI" for PKI.

Figure 1 describes the example of the message format which is based on SAML as the language identifier and Kerberos as the security mechanism.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
                  xmlns:xsd=http://www.w3.org/2001/XMLSchema
                  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
  <soapenv:Header>
    <ns1:Saml xmlns:ns1=http://www.gatewayportal.org/sign.xsd>
      <ns1:SignedAssertiion> encrypted assertion message</ns1:SignedAssertion>
      <ns1:SecurityMechanism>Kerberos</ns1:SecurityMechanism>
    </ns1:Saml>
  </soapenv:Header>
  <soapenv:Body>
    <ns2:SignedBody xmlns:ns2=http://www.gatewayportal.org/signbody.xsd>
                    encrypted SOAP body message
    </ns2:SignedBody>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 1. Message format for secure Web services

3.3. Message-level security infrastructure

The previous section addressed the problem of adapting to multiple mechanisms and security formats. We next must describe mechanisms for actually creating the secure messages and assertions. Our prototype authentication system for using Kerberos and SAML is illustrated in Figure 2 and works as follows:

In client-side process, we developed utility classes using SAML java objects for creating assertions. Specific assertions are created using the appropriate security mechanism. SAML assertions are marshaled back and forth between Java and SAML XML representations. The procedures for generating a secure client message are as follows:

1). A user logs in through a web browser and gets the authentication and a Kerberos ticket on the User Interface Server (UIS).
2). As the user's request, UIS chooses the security mechanism and security markup language. It then establishes the security context with the server for getting the shared key. For example, the Kerberos client on the UIS creates a client session object that contacts a Kerberos server in a session object. The client and server then establish a GSS [26] context, which is maintained on each server in user sessions. Each of these objects possesses one half of the symmetric key pair for a particular user.
3). The Assertion Generator on UIS generates a particular user assertion, for example, user's SAML security assertion.
4). The Signature Generator on UIS signs the user assertion and message (a SAML assertion in SOAP header and SOAP Body messages).
5). UIS rebuilds the SOAP message which is described by Section 3.2 to include the additional security information in the header.
6). Finally, the SOAP request is sent to the desired SOAP server which provides computing services.

In server-side process, SOAP messages are extracted and parsed into several parts such as user's assertion, security mechanism, security markup language, and SOAP body message.

1). The SOAP server establishes a security context with the client for getting the shared session key that will be used for

unwrapping the secure messages. It does not check the signature of the request directly and instead forwards to the authentication service (see [15]), which verifies the signature. For example, the Kerberos server responds positively or negatively to the SOAP server, which may then fulfill the client's request.

2). The SOAP server handles the incoming SOAP request message. It extracts the secure assertion message for the SOAP header, the secure body message from the SOAP body, and the security mechanism name such as Kerberos, PKI, and the security message format such as SAML, WS-security.

3). The SOAP server checks the message format and the security mechanism. It then uses the Unwrap Generator class, which includes the server key object on SOAP server, decrypts a signed SAML assertion and SOAP body message.

4). The validity of the message is checked. For example, SAML fields such as issuer name, "conditions" time limit, subject name, and authorization may all be verified.

5). If the message is determined to be valid, the SOAP server reconstructs the original SOAP messages (removing the security header) and passes this to the server's SOAP engine for processing.

This system illustrates the basic client-server interaction and can be used to build a single sign-on system, such as we describe in [15]. As one example, using this infrastructure, we have developed the specialized use for SAML and Kerberos. We are currently also developing PKI based authentication system.

As the technical resources for our implementation, we have used the SOAP engine, Apache Axis 1.0 [27] which is modified for adding the security process, SAML schema (draft-sstc-schema-assertion-27.xsd) [28], Kerberos, Version 5, release 1.2.2 [9], and Java 1.4 or higher.

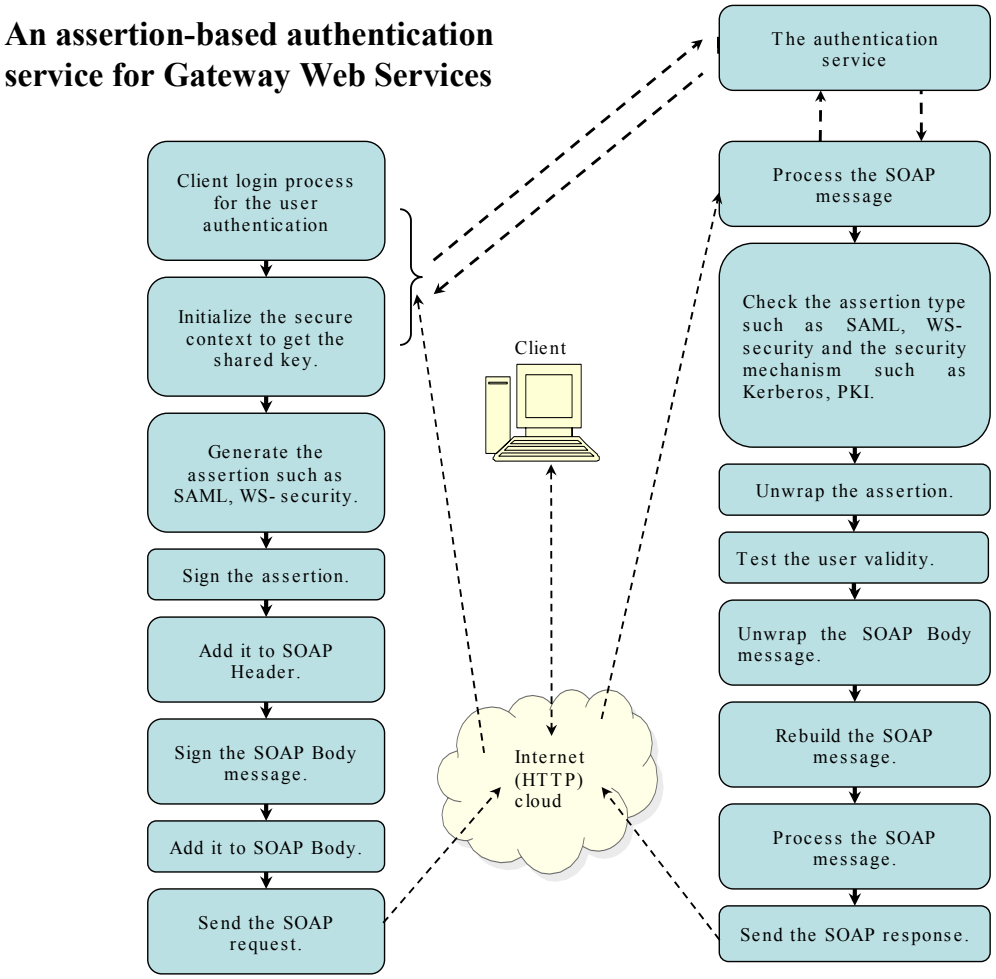

Figure 2. An assertion-based security infrastructure

4. Summary and Future Directions

We have described application-specific services and data models that can be used to encapsulate entire applications independently of the portal implementation for providing services that may be shared among different portals in Ref [8]. In this paper, we have presented the design and implementation of secure computing Web services needed for constituting the computing portal, forming the service-oriented computing architecture. Based on this message-level security system, the portal developer can construct securely specific implementations and composites of primitive service components.

Following our initiation of this project, the National Science Foundation awarded a National Middleware Initiative grant to establish a reference implementation of SAML in Java and C++. This open source reference implementation is available from [29]. We intend to reconcile our implementation with this source.

Web service security is one aspect of our overall program for building Grid Computing Environments. Other topics we are investigating include Web service negotiation to determine quality of service and application and data model services for use in earthquake modeling and prediction.

5. References

1. Foster, I., and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. In *The Grid: Blueprint for a New Computing Infrastructure*, Foster I., and Kesselman, C.,eds. Morgan Kauffmann, 1999.
2. *Concurrency and Computation: Practice and Experience,* forthcoming special issue on Grid Computing Environments. Abstracts and links to accepted papers available from http://aspen.ucs.indiana.edu/gce/gce2001index.html.
3. Vaughan-Nichols, Stephen J.. Web Services: Beyond the Hype. IEEE Computer. Vol 35, No. 2, p 18-21
4. Al Saganich. Java and Web Services Primer. 2001, accessed from http://www.onjava.com/pub/a/onjava/2001/08/07/webservices.html.
5. Fox, G. C., Gannon, D., and Thomas, M. A Summary of Grid Computing Environments. Concurrency and Computation: Practice and Experience, Vol. 14, No. 13-15, pp 1035-1044.
6. Geoffrey Fox, Hasan Bulut, Kangseok Kim, Sung-Hoon Ko, Sangmi Lee, Sangyoon Oh, Shrideep Pallickara, Xiaohong Qiu, Ahmet Uyar, Minjun Wang, Wenjun Wu. Collaborative Web Services and Peer-to-Peer Grids. Presented at 2003 Collaborative Technologies Symposium (CTS'03).
7. Pierce, M., Youn, C., and Fox, G. Application Web Services. Available from http://www.servogrid.org/slide/GEM/AWS.doc. Schemas are also available from http://www.servogrid.org/GCWS/Schema/index.html.
8. C. Youn, M. Pierce, and G. Fox, "Building Problem Solving Environments with Application Web Service Toolkits" Accepted for publication in the ICCS 2003 Workshop on Complex Problem Solving Environments for Grid Computing.
9. B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, 32(9):33-38. September 1994.
10. S. Carmody, "Shibboleth Overview and Requirements." (2001). Available from http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html.
11. R. Clarke, "Conventional public key infrastructure: An artefact ill-fitted to the needs of the information society." In *Proceedings of European Conference on Information Systems (ECIS)*, June 2001. Available from http://www.anu.edu.au/people/Roger.Clarke/II/PKIMisFit.html.
12. Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. A Security Architecture for Computational Grids. Proc. 5th ACM Conference on Computer and Communications Security, pp. 83-92, 1998.
13. Foster, I., et al. The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Accessed from http://www.globus.org/research/papers/ogsa.pdf.
14. Pierce, M. E., Youn, C., and Fox, G. C., "The Gateway Computational Web Portal." *Concurrency and Computation: Practice and Experience*, Vol. 14, No. 13-15, pp 1411-1426 (2002).
15. M. Pierce, et al, "Interoperable Web Services for Computing Portals." In *Proceedings of Supercomputing 2002*. Baltimore, 2002.
16. "Web Service Description Language (WSDL) 1.1", E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. W3C Note 15 March 2001. Available from http://www.w3c.org/TR/wsdl.
17. "Simple Object Access Protocol (SOAP) 1.1", D. Box, et al. W3C Note 08 May 2000. Available from http://www.w3.org/TR/SOAP/.

18. K. Ballinger, P. Brittenham, A. Malhotra, W. A. Nagy, and S. Pharies, "Specification: Web Service Inspection Language (WS-Inspection) 1.0." (2001). Available from http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html.
19. T. Bellwood, et al. "UDDI Version 3.0 Published Specification, 19 July 2002" (2002). Available from http://uddi.org/pubs/uddi-v3.00-published-20020719.htm.
20. GSI SOAP, accessed from http://doesciencegrid.org/Grid/projects/soap/
21. P. Hallam-Baker and E. Maler, eds, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)." Committee Specification 01, 31 May 2002. Available from http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf.
22. "Web Services Security (WS-Security)", B. Atkinson, et al. Version 1.0 05 April 2002. Available from http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.
23. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., and Essiari, A. Certificate-based Access Control for Widely Distributed Resources. Proceedings of the Eight Usenix Security Symposium. 1999.
24. Akenti: Distributed Access Control. Accessed from http://www-itg.lbl.gov/Akenti/.
25. The Castor Project, accessed from http://castor.exolab.org/.
26. Generic Security Services Application Program Interface, Version 2, accessed from http://www.rfc-editor.org/rfc/rfc2078.txt.
27. Apache Axis, accessed from http://xml.apache.org/axis/.
28. SAML schema, accessed from http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-27.xsd
29. "OpenSAML—An Open Source Security Assertion Markup Language Implementation.", accessed from http://www.opensaml.org/.