# Secure Web Services with Globus GSI and gSOAP

Giovanni Aloisio[1], Massimo Cafaro[1], Daniele Lezzi[1], and Robert van Engelen[2]

[1] High Performance Computing Center
University of Lecce/ISUFI, Italy
giovanni.aloisio@unile.it
massimo.cafaro@unile.it
daniele.lezzi@unile.it
[2] Computer Science Department
Florida State University, USA
engelen@cs.fsu.edu

**Abstract.** In this paper we describe a plug-in for the gSOAP Toolkit that allows development of Web Services exploiting the Globus Security Infrastructure (GSI). Our plug-in allows the development of GSI enabled Web Services and clients, with full support for mutual authentication/authorization, delegation of credentials and connection caching. The software provides automatic, transparent transport-level security for Web Services and is freely available.

## 1 Introduction

Recently, the Web Services framework [1] has gained considerable attention. Based on XML (Extensible Markup Language) [2] technologies like SOAP (Simple Object Access Protocol) [3], WSDL (Web Services Description Language)[4], WSFL (Web Services Flow Language) [5] and UDDI (Universal Description, Discovery and Integration) [6], the Web Services approach to distributed computing represents the latest evolution supporting the creation, deployment and dynamic discovery of distributed applications. As a matter of fact, the Internet and the Web allow to publish and retrieve documents easily, and to access a number of commercial, public and e-government services. However, the focus on the usage of such services is now been shifted from people to software applications.

The Web Services framework makes this shift possible, due to the convergence of two key technologies: the Web, with its well known and universally accepted set of standard protocols for communication, and Service-Oriented computing where both data and business logic is exposed through a programmable interface e.g. CORBA (Common Object Request Broker Architecture), Java RMI (Remote Method Invocation), DCE RPC (Remote Procedure Call). Web Services can be accessed through the HTTP (Hyper Text Transfer Protocol) and HTTPS (Secure Hyper Text Transfer Protocol) protocols and utilize XML to exchange data. This implies that Web Services are independent of platform, programming language and network infrastructure.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 421–426, 2003.

Even in the Grid community, the focus is now shifted from protocols [7] to Grid Services [8], as envisioned by the Open Grid Services Architecture (OGSA) [9]. Grid Services extend the Web Services framework, and the Grid itself becomes an extensible set of Grid Services that may be aggregated to provide new capabilities. However, a Grid Service is "a (potentially transient) stateful service instance supporting reliable and secure invocation (when required), lifetime management, notification, policy management, credential management, and virtualization" [8]. So, Grid Services leverage both WSDL and SOAP but additional interfaces able to manage service lifetime, policies and credentials, and to provide support for notification are mandated by the OGSA specification.

Since the OGSA specification is not yet completed and the Globus Toolkit v3 is not yet available to develop production software (only an alpha version has been released as of this writing), we decided to adopt the Web Services framework jointly with the Globus Toolkit v2 as our middleware/computing infrastructure in the GridLab project [10].

The adoption of Globus GSI [11] as the security infrastructure and of the gSOAP Toolkit [12] for the development of Web Services led us to write the GSI plug-in for gSOAP, needed to secure the Web Services developed in the context of the GridLab project. The plug-in provides automatic, transparent transport-level security for Web Services and is freely available [13].

The paper is organized as follows. Section 2 describes the gSOAP Toolkit. We present our GSI plug-in in Section 3, recall related work in section 4 and conclude the paper in Section 5.

## 2 The gSOAP Toolkit

The gSOAP toolkit is a platform-independent development environment for C and C++ Web services. The toolkit provides an easy-to-use RPC compiler that produces the stub and skeleton routines to integrate (existing) C or C++ applications into SOAP/XML Web services. A unique aspect of the gSOAP toolkit is that it automatically maps native C/C++ application data types to semantically equivalent XML types and vice versa. This enables direct SOAP/XML messaging by C/C++ applications on the Web. As a result, full SOAP interoperability can be achieved with a simple API relieving the user from the burden of SOAP details, thus enabling him or her to concentrate on the application-essential logic. The toolkit uses the industry-standard SOAP 1.1/1.2 and WSDL 1.1 protocols and offers an extensive set of features that are competative to commercial implementations, including stand-alone HTTP server capabilities, Zlib compression, SSL encryption, and streaming direct internet message encapsulation (DIME) attachments. For many companies, gSOAP has proven an excellent strategy for developing Web services based on C and C++ applications. For example, gSOAP is integrated in the IBM alphaWorks Web Services Tool Kit for Mobile Devices (WSTKMD) [14]

The gSOAP toolkit was designed with ease-of-use in mind. It exploits a novel schema-driven XML parsing technique to deserialize C/C++ application data

from SOAP/XML messages in one sweep, thereby eliminating the overhead that is incurred by SOAP/XML software with multi-tier communication stacks.

gSOAP is available for download from SourceForge [15] and is licensed under the open source Mozilla Public License 1.1 (MPL1.1).

## 3 The GSI Plug-in for gSOAP

Our plug-in exploits the modular architecture of the gSOAP Toolkit that enables a simple extension mechanism of gSOAP capabilities. To take advantage of a plug-in, a developer must register it with gSOAP, so that full access to run-time settings and function callbacks is granted. The registration associates the plug-in's local data with gSOAP run-time and is done using the gSOAP *soap_register_plugin* function, supplying as one of the arguments the plug-in initialization function. In our case, we perform the necessary initialization steps inside the *globus_gsi* function : we activate the Globus Toolkit I/O module, set-up local data and extend gSOAP capabilities overriding gSOAP function callbacks.

The last initialization step is to provide two callbacks that will be used by the gSOAP environment respectively to copy and delete the plug-in's local data (when de-registering the plug-in). The plug-in's local data can be accessed through the *soap_lookup_plugin* function. Currently, as local data we have Globus I/O related variables (connection attribute, handle, etc), the distinguished names that identify a client or a server when mutual authentication is performed, and the pathname where on the local file system the proxy that a client sends when performing delegation of credentials is written by our plug-in. Finally, we have a boolean variable that distinguishes a client from a server. This mechanism is exploited for instance in the Globus I/O authorization callback that a developer must provide to perform authorization upon authentication: if the software acts as a client, then authorization is based on the server's identity (distinguished name as found in the X509v3 certificate) and vice-versa.

We now briefly describe how the plug-in functions utilize the Globus Toolkit I/O API to provide gSOAP with automatic, transparent transport-level security.

The *gsi_connect* function sets the client mode and calls *globus_io_tcp_connect* to establish a connection to the remote Web Service; the *gsi_disconnect* function checks the state of the connected handle and closes the connection calling *globus_io_close*; the variables related to the server's or client's identity that have been dynamically allocated, are then freed.

The *gsi_send* function is in charge of actually sending data on the Globus I/O connected handle; this is done calling in a loop until needed *globus_io_write*. Symmetrically, the *gsi_recv* function reads data from a Globus I/O connected handle calling *globus_io_read*.

The *gsi_listener*, *gsi_listen* and *gsi_accept* functions are all needed to develop a server; *gsi_listener* sets server mode and calls *globus_io_tcp_create_listener* to create a Globus I/O listening handle, while *gsi_listen* calls *globus_io_tcp_listen* and blocks waiting for incoming connections on the listening handle. Thus, the

Globus function behaves differently from the traditional TCP sockets API *listen* call. Finally, the *gsi_accept* function calls *globus_io_tcp_accept* and in case of success creates the connection handle and calls *globus_io_tcp_get_remote_address* to retrieve the peer's IP address and port.

The *gsi_connection_caching* and *gsi_reset_connection_caching* are used respectively to setup and reset connection caching. This is achieved using the gSOAP internal mechanism for keep-alive connections and calling the Globus function *globus_io_attr_set_socket_keepalive* as needed.

The other functions we provide are needed to setup properly the GSI channel: the developer is allowed to setup the TCP socket reuse addr option, which is useful server side, and to setup authentication, channel mode, protection mode, authorization mode and delegation mode.

Both clients and servers developed using our plug-in can use the Globus I/O authorization callback mechanism; this entails writing a related Globus function called *globus_io_secure_authorization_callback* to enforce the developer's policy.

The plug-in software requires GNU autotools (autoconf v2.57, automake v1.7.2), the Globus Toolkit v2.x and the gSOAP Toolkit v2.2.3d. We provide in our distribution example servers and their related clients, that show how to write a simple threaded server, a pre-threaded server, a simple fork server and a pre-forked server. The threaded servers provide a good example of how to do mutual authentication/authorization and how to setup and use connection caching; the fork servers in addition show how to do delegation of credentials: the server receives from the client the delegated credentials and uses them to submit a job to a remote Globus gatekeeper. Our software is licensed under the open source GNU General Public License.

## 4 Related Work

A similar GSI plug-in for gSOAP has been developed in the context of the Globus project for a C based preliminary version of OGSA. This software provides basically the same functionalities described here, however, our plug-in has been made available to the Grid community before and it is widely used in several academic institutions and software companies, while the Globus plug-in will not be considered production-level software until the final official release of the Globus Toolkit v3. Moreover, the new version of the Globus gSOAP plug-in, to be included in the C based alpha version of OGSA, will support in addition to transport-level security a new GSI mechanism for securing grid services, using message-level security.

The issues that arise using transport-level security are the high computational cost (the entire message must be encrypted and/or signed), the problem related to firewalls (non privileged ports must be opened), and the problem of intermediate hops. A transport-level security mechanism provides two-way encryption and one-way or two-way authentication between a predefined pair of SOAP message endpoints, but SOAP request and response messages may be required to traverse multiple hops between the Web service and the consuming

application. In this case, message security and user identity can be compromised at the intermediate hops.

A message-level security mechanism is safer because there is no exposure at intermediaries (hop-to-hop vs end-to-end) and flexible because building only on proven Web infrastructure (HTTP etc). Another advantage is that firewalls are not an issue here (Web protocols are usually allowed through a firewall) and the associated computational cost may be relatively low, due to the possibility of selective encryption and/or signature of SOAP messages (only some XML elements actually need encryption because they contain sensitive information).

Web Services security is an active research field, and many specifications are now available. We are now developing a complementary version of our plug-in to support message-level security. Both our software and the Globus based plug-in will harness the following specifications:

- WS Secure Conversation [16];
- WS Security [17];
- XML Encryption [18];
- XML Signature [19].

SOAP messaging is enhanced using message integrity, confidentiality and authentication. These specifications allow exchanging a security context between a Web Service and a client, establishing and deriving session keys, and provide a general purpose mechanism for associating security tokens with messages. An important property of these specifications is that they are extensible, i.e., they support multiple security token formats.

## 5 Conclusions

In this paper we have reported about a GSI plug-in for the gSOAP Toolkit. Our plug-in allows the development of GSI enabled Web Services and clients, with full support for mutual authentication/authorization, delegation of credentials and connection caching. The software is being used in several academic institutions for grid projects and testbeds including the GridLab project, moreover, some software companies have demonstrated an interest for it. The plug-in provides automatic, transparent transport-level security for Web Services and is freely available.

We will continue to support the GSI plug-in to provide transport-level security for Web Services; in addition our focus is currently on the development of a complementary version of the software to provide message-level security.

**Acknowledgements.** We gratefully acknowledge support of the European Commission 5th Framework program, grant IST-2001-32133, which is the primary source of funding for the GridLab project.

## References


1. Kreger, H.: Web Services Conceptual Architecture WSCA 1.0. IBM, 2001
2. XML specification. http://www.w3.org/XML/Core/
3. SOAP specification. http://www.w3.org/TR/SOAP/
4. WSDL specification. http://www.w3.org/TR/wsdl
5. WSFL specification. http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
6. UDDI specification. http://www.uddi.org/specification.html
7. Foster, I., Kesselmann, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal Supercomputer Applications, Vol.15, 2001, No. 3, pp. 200–222
8. Foster, I., Kesselmann, C., Nick, J., Tuecke, S.: Grid Services for Distributed System Integration. Computer, Vol. 35, 2002, No. 6, pp. 37–46
9. Foster, I., Kesselmann, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration. Technical Report for the Globus project. http://www.globus.org/research/papers/ogsa.pdf
10. The GridLab project. http://www.gridlab.org
11. Foster, I., Kesselmann, C., Tsudik G., Tuecke, S.: A security Architecture for Computational Grids. Proceedings of 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.
12. Van Engelen, R.A., Gallivan, K.A.: The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. Proceedings of IEEE CCGrid Conference, May 2002, Berlin, pp. 128–135
13. Cafaro, M., Lezzi, D., Van Engelen, R.A.: The GSI plugin for gSOAP. http://sara.unile.it/∼cafaro/gsi-plugin.html
14. IBM alphaWorks, Web Services Tool Kit for Mobile Devices, http://www.alphaworks.ibm.com/tech/wstkMD
15. The gSOAP Toolkit. http://gsoap2.sourceforge.net
16. WS Secure Conversation specification. http://www-106.ibm.com/developerworks/library/ws-secon/
17. WS Security specification. http://www.oasis-open.org/committees/wss/
18. XML Encryption. http://www.w3.org/Encryption/2001/
19. XML Signature. http://www.w3.org/Signature/