SEAMLESS AND SECURE AUTHENTICATION FOR GRID PORTALS

Jean-Claude Côte, Mohamed Ahmed, Gabriel Mateescu, Roger Impey
National Research Council, 1200 Montreal Rd, M50, HPC, Ottawa ON, K1A 0R6 Canada

Darcy Quesnel
Canarie Inc., 110 O'Connor St., Ottawa, ON, K1P 5M9 Canada

Keywords: grid portal, security, web services, single sign-on, digital certificates, secure shell

Abstract: Grid portals typically store user grid credentials in a credential repository. Credential repositories allow users to access Grid portals from any machine having a Web browser, but their usage requires several authentication steps. Current portals require users to explicitly go through these steps, thereby hindering their usability. In this paper we present intuitive and easy to use tools to manage certificates. We also describe the integration of Grid Security Infrastructure authentication into a Java-based SSH terminal tool. Based on these tools, we build an innovative portal authentication mechanism that enables transparent delegation of credentials between clients, grid portal and the credential repository.

1 INTRODUCTION

Grid computing (Foster, 1997] supports sharing of distributed resources across authorization domains. A portal to such a grid allows scientists to securely and easily access grid resources at various locations by using a web browser. For example, users can execute an application, download or upload data and monitor the status of a computing job (Grimshaw, 1997] through a browser. In addition to user mobility, browser-based access provides an easy-to use interface to the grid. Some available grid portal frameworks are GridPort (GridPort], GridSphere (GridSphere] and Unicore (UNICORE]. In this paper, we discuss ways to extend and enhance Grid portals by using client side Java applications (Joy, 2000] that facilitate credential management. These mechanisms were developed in the context of a Grid Infrastructure project in National Research Council Canada.

Typically, grid portals do not run on the client's system. For example, in a Globus Toolkit grid infrastructure based on C-language components, users have to rely on Unix-like platforms to generate certificate requests and do certificate housekeeping by way of command-line tools. The Globus Commodity Grid (CoG) CoG kit (COG] provides Java version of most command-line tools, thereby enabling credential management on all Java platforms. However, the (CoG) kit needs to be downloaded and deployed by the user. It is also missing some functionality present in the command-line tools and does not provide a graphical user interface, as a grid portal user would expect. In this paper we will discuss the design and implementation of some of the missing tools and the deployment of the Java applications using Java Plugin and Web Start technologies. Henceforth, we call the tools that we contribute the Credential Management Tools. Another contribution described here deals with providing a way for grid portal users to open a shell at runtime to a grid resource, which is sometimes required for debugging/inspection of processes on remote resources but is not typically supported by grid portals.

The work presented in this paper is related to the Grid Canada project. The paper is organized as follows. In section 2, we present the architecture of an open-source grid portal with improved authentication and interactive execution services. In Section 3, we describe the certificate management tools, followed by the description of the GSI-based SSH java client in Section 4. Section 5 explains the components and usage scenario of our grid logon client applet and web service. Section 6 contains concluding remarks.

Côte J., Ahmed M., Mateescu G., Impey R. and Quesnel D. (2005).
SEAMLESS AND SECURE AUTHENTICATION FOR GRID PORTALS.
In *Proceedings of the First International Conference on Web Information Systems and Technologies*, pages 140-145
DOI: 10.5220/0001229501400145
Copyright © SciTePress

2 GSI-BASED GRID PORTAL SERVICES

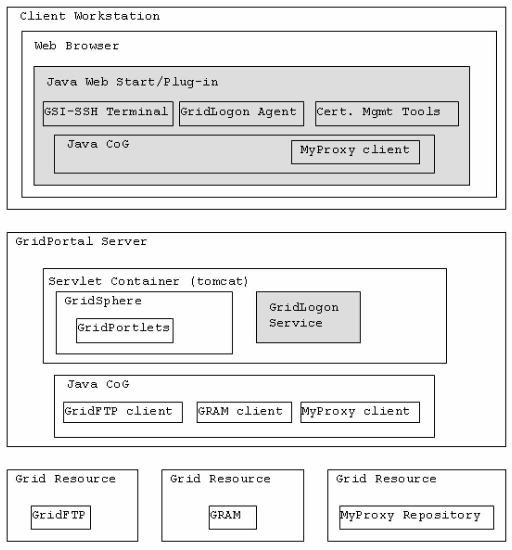


Figure 1: Grid Portal Architecture and Components

The problem with grid portals that we tackle in this paper is that the client needs to be aware to certain extent with the grid authentication mechanisms and operations. Even with the use of grid MyProxy server as a third entity for maintaining proxy certificates, the user should go through several steps to upload his credentials, renewing it if necessary and destroying it when he opts to. This hinders the usability of the use of credential repositories and making them harder to use for novice and non-computer savvy users. We try to solve those issues through the addition of new and useful automatic tools (we called them grid-logon services and protocols) to manage certificates more easily and transparently especially against grid portals. Moreover, we integrated these new modules and protocols as a new useful feature to other client tools such as Java-based SSH terminal clients. This would provide very powerful portable service for end-user during the grid portal session.

In this section we show the design of the Credential Management Tools, and then we show how we added a GSI Authentication module to the SSHTerm Java application. Finally, we present a novel Grid Portal Logon Mechanism that enables transparent single sign-on. Figure 1 depicts an overview diagram of typical Grid Portal layout. Our new modules, which we will discuss later in this paper, are highlighted in grey. We based our portal implementation and protocols on GridSphere grid portal (GridSphere].

3 CERTIFICATE MANAGEMENT AND AUTHENTICATION MECHANISMS

The Public Key Infrastructure (PKI) is used by most grid middleware. For example Globus uses GSI which in turn is based on PKI. Under PKI, a user generates a private/public key pair using a certificate request tool; then the user sends his/her public key to a certificate authority (CA). The CA verifies the user's information and provenance. The CA uses its private key to sign the user's public key and grants the user a digital security certificate. The user receives this certificate from the CA. The private key along with the signed certificate are the user's credentials, which can be used to authenticate and authorize access to grid resources. Similarly, an administrator requests a certificate on behalf of the grid portal entity and for every grid resource in general.

A grid user could access grid resources using his/her credentials by way of the portal. Thus, user's credentials must be delegated to the portal since it is the portal that accesses the grid resources on behalf of the user. This delegation is typically done using a credential repository (for example, a MyProxy (Novotny, 2001] repository). Here's a diagram (Figure 2) showing several ways of GSI authentication between a grid client and a grid resource.

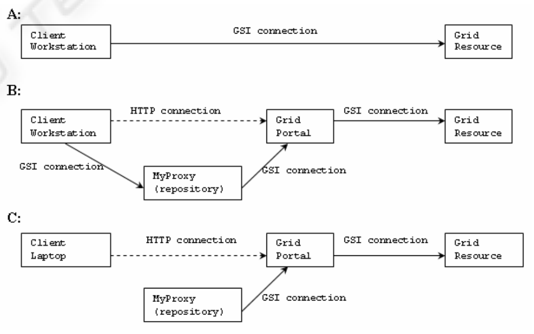


Figure 2: Typical usage of MyProxy repository by grid portals.

-Scenario A: A command line tool is employed by the user to send a job request to a grid resource. Delegation is done using the GSI mechanism.

-Scenario B: A command line tool is used to delegate his/her credential to a credential repository, using the GSI mechanism. The user then uses the portal to send the job request over plain HTTP or HTTPS. The portal retrieves the user's credentials from the repository on behalf of the user and sends the job request to the grid resource over GSI connections.

-Scenario C: Assuming valid credentials are already stored in the MyProxy repository, a grid portal user simply uses a browser to send a job request from any location.

During development of our Grid Canada portal (GridCanada] we followed this approach but quickly found that pieces were missing to make this process intuitive and easy to use. In addition, there were no tools available to Windows and Mac users to generate certificate requests and renewals. So these users had to first find a Linux machine with Globus installed on it to generate their certificate requests and move their private keys and certificates back to their favourite operating system, a major security faux pas. Hence we implemented both of these command line tools in Java and thus could be run by any Java-enabled browsers. We have contributed them to the Globus Java CoG kit. While this Globus COG kit provided portable certificate management, users were still required to install the Java CoG kit and execute command line tools. So the logical next step was to make this process truly transparent by (i) providing a graphical user interface; (ii) supporting "point-and-click" installation of the Java code. So we developed our credential management tools; a set of Java applications deployed using Web Start and Java Plug-in technologies.

Both the Java Plug-in and Web Start technologies employ code-signing; and PKI to ascertain the provenance and integrity of the application being installed. The code is bundled into Jar files and signed using the Grid portal's private key. The Jar files contain a digital signature (an encrypted hash of the Jar file itself) and a certificate chain. A certificate chain is a list of hierarchically ordered public-key certificates and it is used to determine the ultimate signing authority of a certificate.

Web Start technology can verify the certificate chain all the ways up to its root CA certificate. It checks if the root CA certificate is contained in the database of trusted root CA certificates. Web Start comes already installed with well-known root CA's such as VeriSign Inc. and Entrust Inc. Since our Grid Canada root CA is not contained in the database it needs to be manually installed once by the user.

The certificate management tools described in this paper support a comprehensive set of operations: generating a certificate request, a host certificate request, and a certificate renewal; viewing a local certificate; revoking a certificate; delegating credentials to a MyProxy repository. All these tools are intuitive, easy to use and self-installed using Java deployment technologies.

4 GSI-SSHTERM AUTHENTICATION MODULE

This section describes GSI-SSHTerm, a pure portable Java application (as shown in figure 3) that enables users to open remote shells on grid resources. Shell access to grid resources is sometimes required to debug and inspect remote processes running on grid resources. As mentioned in the introduction, we did not find any Java SSH terminal able to authenticate using the GSI authentication mechanism. So we designed and implemented our own GSI authentication module and integrated it into the SSHTerm tools from the SSHTools project (SSHTools].

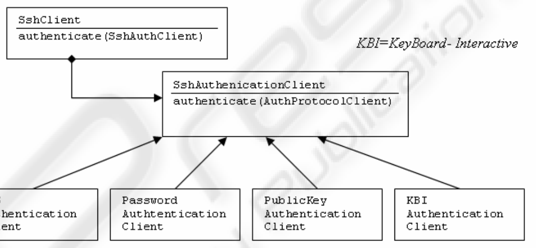

SshClient
authenticate(SshAuthClient)
KBI=KeyBoard- Interactive
SshAuthenicationClient
authenticate(AuthProtocolClient)
GSS Authentication Client
Password Authtentication Client
PublicKey Authentication Client
KBI Authentication Client

Figure 3: SshAuthentication class hierarchy diagram.

Until now, the Globus Toolkit uses the GSI to enable secure authentication and communication over an open network. GSI provides a number of useful services for Grids, including mutual authentication and single sign-on. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation. The Globus Toolkit's implementation of the GSI adheres to the GSSAPI (RFC2078].

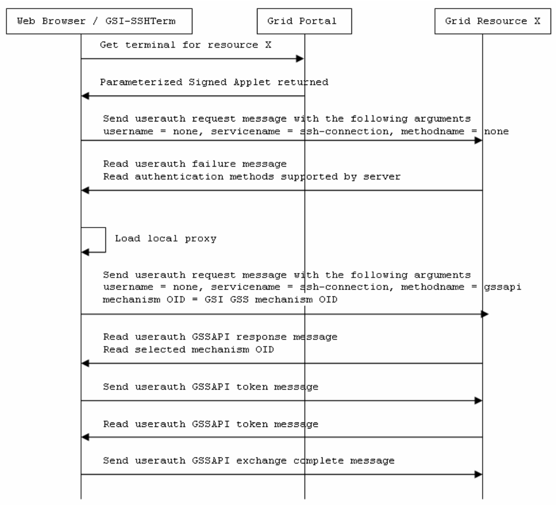
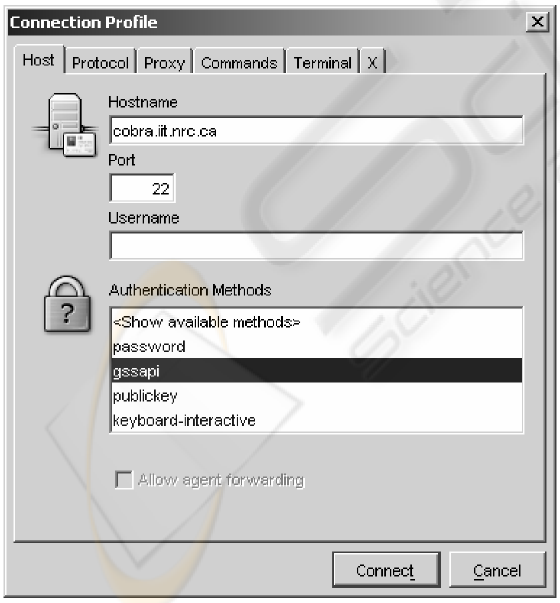

Web Browser / GSI-SSHTerm
Grid Portal
Grid Resource X
Get terminal for resource X
Parameterized Signed Applet returned
Send userauth request message with the following arguments username = none, servicename = ssh-connection, methodname = none
Read userauth failure message
Read authentication methods supported by server
Load local proxy
Send userauth request message with the following arguments username = none, servicename = ssh-connection, methodname = gssapi mechanism OID = GSI GSS mechanism OID
Read userauth GSSAPI response message
Read selected mechanism OID
Send userauth GSSAPI token message
Read userauth GSSAPI token message
Send userauth GSSAPI exchange complete message

Figure 4: GSI-SSHTerm Interaction diagram.

Our GSI authentication implementation conforms to the architecture of SSHTools' SSHTerm application and to the IETF specification "GSSAPI Authentication and Key Exchange for the Secure Shell Protocol" (GSSAPI]. The SSHTerm application provides password, public key and keyboard interactive authentication mechanisms. We have added a GSI authentication module to it.

Now we will describe a grid portal session, which is shown in figure 4. This scenario is for a user that connects to the portal from an arbitrary machine (e.g., a laptop) on which the user's credentials are not installed.

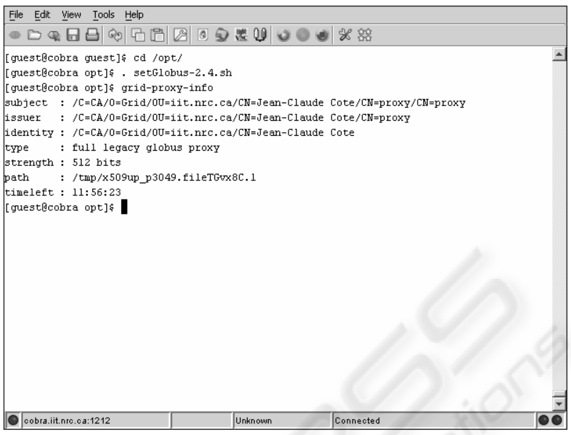
The user logs into the Grid Canada Portal, which brings up a list of grid resources in the infrastructure. The user could select to login any one resource of the list. A GSI-SSHTerm application, as shown in figure 5, is automatically downloaded and deployed on the client (if necessary) and then starts automatically. The application starts having the grid resource's URL as an input parameter. When the application starts, the authentication process would automatically detect that the user is using a terminal device with no local credentials and thus it prompts the user for his/her MyProxy account name and password to retrieve a proxy credential. The retrieved delegated proxy will be used to mutually authenticate with the grid resource using the GSI authentication protocol, which implies credential delegation.



Hostname: cobra.iit.nrc.ca
Port: 22
Username:
Authentication Methods: <Show available methods>, password, [✓ gssapi], publickey, keyboard-interactive

Figure 5: Connection profile dialog of the SSHTerm.

Therefore, the user can use shell commands to the remote resource if needed, as shown in figure 6. He is also able to access other grid resources using the same terminal session since he/she has delegated his credentials on this resource.



```
[guest@cobra guest]$ cd /opt/
[guest@cobra opt]$ . setGlobus-2.4.sh
[guest@cobra opt]$ grid-proxy-info
subject  : /C=CA/O=Grid/OU=iit.nrc.ca/CN=Jean-Claude Cote/CN=proxy/CN=proxy
issuer   : /C=CA/O=Grid/OU=iit.nrc.ca/CN=Jean-Claude Cote/CN=proxy
identity : /C=CA/O=Grid/OU=iit.nrc.ca/CN=Jean-Claude Cote
type     : full legacy globus proxy
strength : 512 bits
path     : /tmp/x509up_p3049.fileTGvx8C.1
timeleft : 11:56:23
[guest@cobra opt]$ 
```

Figure 6: Java GSI-SSHTerm showing delegated credentials.

5 GRID PORTAL LOGON MECHANISM

In order for the GSI-SSHTerm to be able to retrieve a credential from a MyProxy server, it needs to establish a GSI connection to the server. However, because GSI implies mutual authentication, the client needs to possess a credential. To break this circular dependency, an anonymous credential (i.e., a credential with no distinguish name) can be used.
For this approach to work, the MyProxy server needs to be configured to allow anonymous connections. However, administrators may not allow this type of identification.

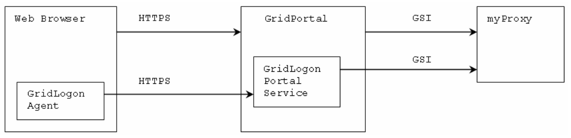


Figure 7: GridLogon Agent/Service interaction diagram.

Another solution would be to delegate the credentials to an application running on the user's local machine through a signed Jar file. The Grid portal already has the user's credentials delegated from a MyProxy server, which it uses to submit jobs on the user's behalf. So one could bundle the user's credentials into a signed Jar file downloaded over an HTTPS connection (HTTPS is HTTP combined with transport layer security (TLS)). This technique should be used cautiously since it involves transmitting the user's private key (via TLS) across the network.

A GSI-SSHTerm application deployed through Web Start is very easy to use however it does require that a user provide his/her MyProxy account name and password. It also assumes that a MyProxy server with anonymous GSI authentication enabled is available and reachable by the client. Getting rid of this extra step altogether would be ideal. The user would simply click on a resource link and a shell would open. To satisfy all of these constraints a new grid logon mechanism is presented. This mechanism involves a "grid logon agent" and "grid portal logon service" pair.

The grid logon agent is an applet that simplifies the grid portal logon process. This applet works in concert with the grid portal logon service as shown in figure 7. The service retrieves a credential and stores it locally. Simultaneously, it would log the user into the portal in the usual way. Then when the user selects a grid resource, the SSHTerm application will load the local credential from the location, which the logon applet saved it.

The grid portal logon service is a secure Web Service hosted by the Grid portal. This service serves as a front end to the MyProxy server and thus only the Grid portal (using its portal credentials) would communicate with the MyProxy server. Since a secure web service is exposed through the well-known HTTPS port it ensures that all users will be able to use it. A grid portal logon service also offers other benefits like constricting the access to the MyProxy server and thus gives the grid portal more control over the use of the MyProxy server. For instance, users would only be able to log into the grid portal logon service using their portal account name. The portal could validate the account name and the Distinguished Name (DN) of the user's certificate. The grid portal logon service could also poll the MyProxy server and detect when a user's delegated proxy is about to expire and notify the user using his email address retrieved from the portal database.

We describe the usage scenario between the grid logon agent and logon service as follows:

1. The user points a web browser at the grid portal welcome page and clicks on the grid logon agent applet. Using the applet, the user enters his/her portal account name and a MyProxy password (should be different from the portal's password). The grid logon agent sends this information to the grid portal logon service over HTTPS.
2. The grid portal logon service checks if there are already deployed credentials for this user on the MyProxy server and if it should be refreshed. In case of success, the grid portal logon service sends back a status indicating success and delegates those credentials to the grid logon agent. In case of failure, the applet is notified that "credentials/refresh is required".
3. If the grid logon agent receives a failed message it tries to locate credentials on the client machine.
3.1. If a local credential exists it prompts the user for the private key password, delegates credentials to the grid portal logon service using a GSSContext [RFC2078] and sends the portal account name.
3.1.1. The grid portal logon service then creates a portal account name and certificate's distinguish name mapping and asks the portal to validate this mapping thus insuring that only the proper user can do MyProxy "put" operation using this portal account name. If all is ok the grid portal logon service then puts those credentials on the MyProxy server and the process goes to step 5.
3.2. If no local credentials can be found, the user is in a situation where he/she will be unable to access grid resources, having no credentials locally and no credentials on the MyProxy server. The user should then be notified of the situation and be invited to log into the grid portal service from a machine with credentials. The user should also be reminded that he/she can simply log into the portal using the portal account name and portal password but be warned that he/she will not be able to access any grid resources.
4. If the grid logon agent applet receives an "ok" it stores the delegated credentials
5. The grid logon agent then generates a URL with the user's portal account name and the MyProxy password as parameters and then tells its parent browser (the browser that started the grid logon agent applet in the first place) to load the URL.
6. The portal uses its Single Sign-on mechanism to authenticate the user using the portal account name and MyProxy password. Simultaneously it retrieves the user's credentials from the MyProxy server.

This grid logon mechanism enables ubiquitous credential delegation in the sense that credentials are delegated between the client with or without certificates and the Grid portal. This mechanism is also transparent to the user. It does not put any restrictions on the MyProxy server configuration. It also uses the well-known and usually permitted HTTPS port. It gives the portal more control over what MyProxy account names are used by users.

Finally this mechanism extends the existing portal authentication mechanism and thus does not restrict users from using the conventional form-based portal authentication. Meanwhile the state during the session between the client browser and the GridSphere-based grid portal are maintained through the use of cookies.

6 CONCLUSION

In this paper, we described extensions to the Globus Toolkit and SSHTerm that support seamless credential management for accessing grid resources and support portable secure shell access to grid resources (GSI-SSH terminal). We also introduced an innovative portal authentication mechanism using a grid logon signed applet that would enable client-side Java code to authenticate to grid resources directly using GSI mechanisms. The functionality has been implemented as Java applications that could be deployed with the Web Start and Java Plugin technologies. In future, we expect to do some tests to evaluate the scalability and performance of our solutions against multiple users and resources.

ACKNOWLEDGEMENTS

We would like to thank Michael Russell of the GridSphere project at Albert Einstein Institute for his co-operation in verifying the design of the GridLogon mechanism. His knowledge of Grid portals was valuable. We are also grateful to Jim Basney of the National Center for Supercomputing Applications for his help in debugging the GSI-SSHTerm authentication module. His knowledge of inner workings of the GSI enabled OpenSSH server was very useful.

REFERENCES

COG: Commodity Grid Kits, http://www-unix.globus.org/cog/

Foster, I. and Kesselman, C., 1997. Globus: A metacomputing infrastructure toolkit. The International *Journal of Supercomputer Applications and High Performance Computing*, 11(2), pp. 115–128.

GridCanada, http://www.gridcanada.ca

GridPort, http://gridport.net

GridSphere project, http://www.gridsphere.org

Grimshaw, A., Wulf, W., and the Legion team, 1997. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1), pp. 39–45.

GSI: Grid Security Infrastructure, http://www.globus.org/security/

GSSAPI IETF specification "GSSAPI Authentication and Key Exchange for the Secure Shell Protocol" http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-secsh-gsskeyex-02.txt

JGSS: The org.ietf.jgss package, http://java.sun.com/j2se/1.4.2/docs/api/org/ietf/jgss/package-summary.html

Joy, B., Steele, G., Gosling, J., and Bracha, G., 2000. The Java language specification. *Addison-Wesley, second edition.*

Novotny, J., Tuecke, S. and Welch, V., 2001. An Online Credential Repository for the Grid: MyProxy. *10th IEEE International Symposium on High Performance Distributed Computing, San Francisco CA.*

RFC2078: IETF specification Generic Security Service Application Program Interface, Version 2, http://www.ietf.org/rfc/rfc2078.txt

SSHTools project, http://sourceforge.net/projects/sshtools/

UNICORE project, http://www.unicore.org/