Grid Single Sign-On in CCLRC

Jens Jensen and David Spence and Matthew Viljoen
CCLRC Rutherford Appleton Laboratory

Abstract

This paper presents the latest results in on-going work on developing a single sign-on solution to access Grid resources. Since last year's e-Science All Hands Meeting, we have adapted a Java terminal by integrating it with site authentication infrastructures to provide access to the NGS and CCLRC's SCARF cluster, using MyProxy to manage the certificates and proxies that are essential for Grid access. We describe the architecture and details of the implementation, and how it fits into the site infrastructure, as well as future Shibboleth deployments. Although the work is done at CCLRC, this work is applicable to any site with a Kerberos or Active Directory infrastructure, and will be of interest to anyone working with authentication technologies.

1 Introduction

At last year's AHM we presented plans for implementing Single Sign-On (SSO) to Grid resources at CCLRC, including the NGS. We demonstrated a prototype portal with a simple job management workflow and web-based terminal access.

In this paper we describe our achievements and experiences in providing full terminal access. Thus, this paper focuses mainly on providing *terminal* access, as opposed to *portal* access. CCLRC has made no effort to further develop the portal prototype [1], nor to integrate SSO with existing Grid portals, but the terminal described in this paper is successfully in production. Grid access is integrated with wider site SSO efforts. We briefly describe how this work integrates with other SSO work, and related future directions.

1.1 Background

Many projects claim to have single sign-on. To these projects this means that a user only has to type the password or passphrase protecting their credentials once, or once every day. For example, Globus has the Globus proxy [2], which enables a user to access Globus resources without having to retype the passphrase that protects the user's (X.509) private key. `ssh` has the ssh "agent": a user space daemon which caches the unprotected (ssh) private key in memory.

On the other hand, for two primary customers of CCLRC's internal Grid resources (Diamond and ISIS), SSO means *integrated user management*. They want their user offices to be able to set up accounts for new users, including visiting scientists, and they want their databases to be consistent.

It is thus useful to give a definition of SSO, and we do this in section 1.2.

1.2 Aims and Use Cases

The principal use case for this work are:

1. A user on site logs into the site's Microsoft Active Directory (AD) [3] system – which, for the purposes of this paper, is compatible with Kerberos V [4]. Using this token, or ticket, the user is able to access the Grid.

2. Users who have certificates from the e-Science Certification Authority (CA), or from another Grid CA, can access the Grid resources using their certificates, both on site and off site. For users who have both local (AD) accounts and a certificate, the identity management knows that those two identities belong to the same person. In other words, the user can access the same resources with AD on site and using their personal certificate from a laptop when they're travelling.

3. Finally, for the user who does not have access either to the local account, or to their e-Science certificate, the terminal falls back to asking for username and password. The username and password are both the ones the user would use to authenticate locally.

The conditions were also that the software should be easy to install; for example, a user should be able to install and run it without having privileged access to their own desktop computers.

Finally, since many users were using `ssh` based login, we need to migrate them off that: they must get certificates, the certificates must grant them access to the same resources that they could access via `ssh`, and in particular, their identity must be preserved — i.e., we need to map between the `ssh` public key and the certificate public key.

It is clear that identity management is an essential part of SSO. Moreover, identity management is quite complicated, particularly over longer timescales. We will touch upon the subject briefly in the next section and in the Architecture section below, but otherwise it is outside the scope of this paper.

1.3 Integrating SSO

As mentioned above, the terminal work integrates with other site SSO efforts. The user offices of the CCLRC facilities will manage the user accounts for both local and external users in CCLRC's staff database. The challenge for the facilities is to populate the database with consistent data, because a given user's details may not be up to date, or the user may be present more than once in the database – for example, the user may have registered with more than one experiment.

For our purpose, we need to add the Grid identity information (mainly the Distinguished Name (DN)) to this database. Populating it initially is already a challenge, because 200 personal certificates have been issued to CCLRC and Diamond, and need to be matched to the Active Directory id (or Kerberos name). Strictly speaking, each user must do it for themselves by proving *both* their identities to the database – or, more precisely, an agent running on behalf of the user which is able to ask the user for proof of both identities (or pick up SSO versions of both), contact a server which can pick up both tokens and update the database.

1.4 Other Work

In this section we briefly describe other efforts in the area of SSO. The simplest example of SSO is of the site that is fully "Kerberised," where Kerberos authentication grants access to all resources. This doesn't immediately support roaming users, but it does provides SSO within the site. For Grid use such a site will have to implement *credential conversion* via MyProxy [5] or KCA (which provide credential conversion, generating short-lived certificates for users presenting Kerberos tickets). This is the approach taken by Fermilab. A KCA converts a Kerberos ticket to a short-lived X.509 certificate (similar to a Globus proxy), but these certificates cannot be used at many external sites because KCAs are not fully trusted compared to *classic* CAs [6] (i.e., CAs that issue long-lived end-entity certificates), although this is slowly improving. Another major disadvantage is that this approach does not scale: current Grid middleware needs *all* CAs in all hierarchies installed along with their signing policies and CRL endpoints. If each institution has its own CA the international collaborations would be overwhelmed by the sheer number of CAs. The same problem arises in the trust context: each CA has to negotiate trust with peer CAs and international resource providers, and they would become overwhelmed if there were more than one CA per country. One could argue that it should be sufficient to review the root's policy, but most resource providers need to know details of each individual site's credential conversion policy: who can get an account, how is the DN tracked back to the identity, etc.

Another approach was taken by Brookhaven National Laboratories. Like CCLRC, they were not keen on having disparate `ssh` and Grid key management infrastructures in addition to their site Kerberos infrastructure. They integrated a One Time Password (OTP) system with parts of their site infrastructure [7]; they found it was simple to integrate, but the cost was relatively high since users had to have hardware OTP tokens.

2 Architecture

Figure 1 gives an overview of the architecture of the CCLRC SSO solution. In the left half of Figure 1 the site authentication system is employed, which is the only authentication infrastructure users are concerned with. Depending on the implementation of SSO the site authentication token can be anything from a password to a Kerberos [4] ticket and correspondingly the user may (in the case of passwords) or may not (in the case of a Kerberos ticket) have to be actively involved in supplying the token. From the user's point of view, the difference is whether they have to type their password only once per session (e.g., when logging on to a computer, case 1 from Section 1.2), or whether they have to type it again when accessing the user interface (case 3) — but at least then, it is the *same* username and password as the federal one. In either case, in the architecture, the token is passed on to the MyProxy [8] server by the user interface, which then checks its validity with the site authentication infrastructure.

Case 2 from Section 1.2 is subtly different. From the user's point of view, for users who already have eScience certificates, they must upload an unencrypted proxy to the MyProxy server, typically once every week. This requires authentication, so the MyProxy upload tool must be integrated with SSO.

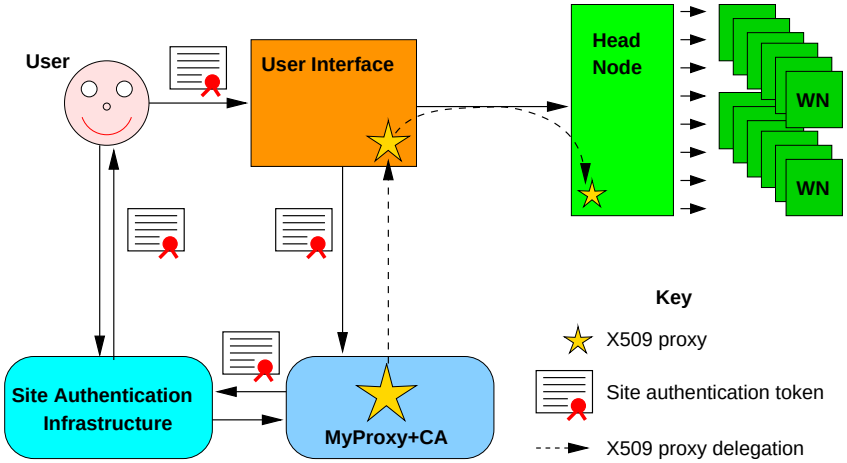

Figure 1: The architecture

They may have a passphrase for the browser's keystore and another one protecting the private key if the credentials are exported from the browser. These passphrases are independent of the SSO infrastructure — indeed, it is possible to leave the browser keystore or the exported key entirely unprotected (of course this violates the CA's policy).

In any case, users need to unlock their credentials, and to authenticate to the MyProxy server, using any method, to upload their proxy. Alternatively the user interface can directly use the local X.509 credentials (either in the browser, or exported) for authentication of the user. In this case the user will have to type the passphrase that unlocks the private key.

Grid access always requires an X.509 credential. For users without UK eScience certificates, we solve the SSO to the Grid problem by generating short-lived, low-assurance (see below) credentials within the MyProxy server.

The right of Figure 1 is within the Grid Security Infrastructure (GSI) domain and uses X.509 certificates [9] and proxies. Within the SSO system described here, the MyProxy server is the source of these credentials. When a user correctly authenticates the MyProxy server will generate a proxy for the user, based on either the user's real UK eScience certificate (if it has been previously uploaded) or a short-term, low-assurance certificate generated by its internal CA. The user interface can then use this proxy to authenticate the user to the head node of a resource and can also further delegate the certificate to the head node to allow other resources to be used.

2.1 Assurance

Assurance, in this context, is a rough measure of the extent to which authentication token can be trusted to represent the user's true, "real life" identity. The Grid community defines four assurance levels [10]:

- *Rudimentary*: there is essentially no *proof* of the user's identity;
- *Basic*: The applicant provides proof of identity to an agent of the issuing authority via a reasonably secure method, but may not have to appear in person;
- *Medium*: The applicant must appear in person to the agent, presenting adequate photographic proof of identity;
- *High*: Essentially as medium, except at the time of renewal, the applicant must present photo id again, and there are greater requirements regarding the protection of the user's private key.

To be internationally approved, the eScience CA has to operate to the medium assurance level. The CA and its verifying agents form a well-defined entity that can account for the flow and subsequent use of personal data throughout, from the first application to the final certificate expiry. In the terminology of the Data Protection Act [11], the CA (more specifically, the CA manager) is the *data controller*: the person who defines which data is necessary and required, what it is used for, and how it should be stored and managed. Furthermore, in the case of a problem such as misuse of a resource, the resource admin can report the user's DN to the CA and the CA can prove, using its personal data, who the user is in "real life."

With a credential conversion service, managing the user's data is *externalised*: it is in the hands of some other authority, external to the issuing authority and its agent — not just at the time of identity verification, but throughout the "lifetime" of the data. It

thus becomes harder for the service to even define, and much less guarantee, its assurance level. Some sites can generate authentication tokens from their payroll databases, but those are the minority. Most sites' databases have visitors and contractors, etc.

Worse yet, in the misuse case mentioned above, suppose a resource admin reports the DN to the credential conversion service admin, but the latter cannot provably tie the site authentication token to the user's real life identity. Only people with appropriate access to the site database (usually HR staff) can do that, and they probably will not, for data protection reasons, because the resource is external to their site. This aspect may make resource administrators less inclined to trust the credential conversion service.

Since we cannot say, even for our own site, what is the assurance defined by the site staff database, we, somewhat arbitrarily, refer to certificates created by the SSO service (from the MyProxy CA) as "low-assurance". The reader may find it helpful to compare it to basic assurance: in some sense, MyProxy is an agent which receives and validates an authentication token from the applicant, and issues a certificate.

Another potential problem with the SSO architecture is that each site has its own credential conversion service. This means that rather than reviewing and installing a single point of trust, the resource admin must now review policies of tens of services, and install them individually.

In theory, if all sites' services could agree to have:

- a common policy of adequate level (the policy will essentially be the "lowest common denominator," i.e., loose enough to accommodate all sites),
- a common root certificate with each site's service CA issued by that root,

then, because these services do not issue CRLs, it should be possible just to trust the common root. However, on the Grid, Globus still requires the signing policy files, to perform extra validation of the namespaces.

Thus, to install a credential conversion CA for each site, may be possible in a single (small-ish) country, or in projects with few collaborating sites, but on a larger, or international, scale it becomes difficult.

For these reasons, and also to provide host and service certificates, an SSO infrastructure cannot remove the need for a national CA. Even for a project the size of NGS, work is required to make future SSO infrastructures compatible; some of this work will be done in the context of the Shibboleth roll-out, but Shibboleth is not yet integrated with the Grid middleware.

3 Implementation

For Grid SSO in CCLRC we have leveraged the Kerberos/Active Directory [3] security system already present to allow users to log in using either a Kerberos token or their site password. This addresses two of the SSO use cases (one and three) mentioned in Section 1.2. The different schemes are employed based on whether or not the user has a valid Kerberos ticket present.

The SSO solution also must support users that have an X.509 certificate locally (e.g., as issued by the e-Science CA). This is the second use case mentioned in Section 1.2, and is addressed by allowing users to access certificates stored in a variety of formats.

The e-Science CA is a web-based CA: when users download their personal certificate, it is stored in the browser, and usually exported from the browser in PKCS#12 format. For Grid work, users would typically have to convert this format into PEM format at the command line, using relatively complex OpenSSL commands. However, this conversion is not required in the SSO project: it is possible to use the certificate in both formats, both PKCS#12 and PEM. It is even possible to use the certificate directly from the user's browser, thus making the exporting stage unnecessary. This once again follows the SSO philosophy, allowing users to authenticate to Grid resources with a minimum of effort.

The main components involved in the CCLRC SSO system are the MyProxy server and the Java GSI-SSHTerm, which forms the user interface component from Figure 1.

3.1 MyProxy Server

The standard MyProxy server distribution comes both with Kerberos (using SASL authentication) and CA support as compile time options. Two MyProxy server instances on the same machine are employed for our SSO solution, sharing the same certificate store. Both also have the build-in CA support enabled, again issuing certificates from the same root certificate. The first is a non-Kerberos MyProxy server (using the normal MyProxy port of 7512); this is linked through a PAM module to the site authentication system for site password access to the SSO facilities. A normal MyProxy server also allows users to upload real e-Science certificate proxies to the server using the standard tools. The second MyProxy server runs on port 7513 and uses the Kerberos/SASL authentication.

When a low-assurance certificate is required the MyProxy servers call out to a small program which is used to map a site user id to a DN. To do this it uses LDAP access to the site Active Directory and constructs the DN as follows: **/C=UK /O=eScienceSSO /OU=CCLRC /UID=***uid* **/CN=***firstName surname*. Resource administrators must explicitly trust the MyProxy CA by importing its root certificate to the resource and adding the SSO DNs of legitimate users to the resource's grid-mapfile.

3.2 Java GSI-SSHTerm

The Java GSI-SSHTerm[1] is used as the user interface in our SSO solution, providing GSI-SSH access to grid resources, through a platform independent Java SSH terminal. The original Java SSH terminal is an open source project hosted on Sourceforge. Jean-Claude Côte at NRC-CNRC then developed a GSI module for the SSH terminal, which we have extended and improved. The GSI module and our additions rely on the Globus Java Commodity Grid (CoG) libraries [12]. For the SSO project the MyProxy parts of the CoG libraries were modified to support the Kerberos based MyProxy servers through the addition of support for the SASL authentication method. The Java GSI-SSHTerm uses Kerberos authentication if the user has a Kerberos ticket and otherwise allows users to supply their password to access the MyProxy servers. However, making use of the Kerberos token requires Java 1.6, and only Java 1.4 and 1.5 are widely available. We have successfully used a beta release of Java 1.6.

Although this new functionality is exposed through the Java GSI-SSHTerm it could be used with any other Java based grid user interfaces and tools such as a proxy upload tool.

In addition, as previously mentioned in Section 3, the Java GSI-SSHTerm allows direct access to real e-Science certificates in a number of ways.

3.3 Renewal

When the terminal's proxy expires, the user is prompted to re-authenticate, either via MyProxy or via the user's local credentials. For longer running sessions where the user stays logged in but is absent from the terminal, command output is *not* lost. The terminal prompts the user for authentication, but continues to display the command output, but the user cannot type anything into the terminal until the authentication is successful. Of course, if the user authenticates with a local Kerberos ticket, it is sufficient that the terminal can pick that up. On Windows, for example, Windows renews the ticket automatically when necessary, and the terminal re-authenticates the user without any user intervention.

The proxy on the remote host, the Grid head node into which the user logs in via the terminal, can also expire. This proxy is created initially by the terminal, but is not renewed by the terminal in the current implementation. The user will have to re-authenticate to the MyProxy server to get a new proxy out of it.

3.4 File transfer

Grid access is more useful if the user can transfer data from and to the Grid head node within the session. The Java GSI-SSHTerm includes a graphical transfer interface that the user can use to upload and download files using the SFTP protocol (over GSI).

3.5 X forwarding and VNC client

The Java GSI-SSHTerm is capable of secure "X forwarding" to an X Window System server running on the same computer as the Java GSI-SSHTerm program. In other words, programs running on the head node can display their windows on the user's desktop machine.

Alternatively, users can use a Java VNC client built into the program to connect to a VNC server session running on the head node.

3.6 CCLRC Changes to the GSI-SSHTerm

To bring the SSHTerm and the GSI module into a form useful for Grid users we have had to both add a number of features and also perform hardening and bug-fixing throughout the code.

Within the main body of the SSHTerm code, changes included new build scripts, large changes to the X forwarding code to allow correct authentication under UNIX and to allow UNIX domain sockets to be used, improvement to protocol conformance in the choice of key exchange, cypher, compression, etc., algorithms (taking the responsibility for this bewildering decision from the users) and a redesign of the open connection dialog to allow easy connection for users connecting to Grid resources using default settings similar to default SSH settings.

In the area of GSI support the original authentication method implemented by Jean-Claude Côte could not authenticate users who did not know their username. To enable logon without a specified username we had to implement the "external-keyx" authentication algorithm. In addition, this authentication algorithm requires the "gss-group1-sha1-*" key exchange algorithm, which we also implemented. Other work in the area of GSI support included porting Jean-Claude Côte's code to a later version of

[1] http://www.grid-support.ac.uk/content/view/81/62/

the CoG kit and implementing methods to obtain proxies from additional sources (MyProxy servers, PKCS#12 files and browsers) and integrating these methods into a consistent and unified user interface.

These changes to the GSI-SSHTerm are in addition to the changes to the CoG libraries and the GSI-SSHTerm to allow Kerberos authentication to MyProxy servers.

3.7 Deployment

As we write this, our SSO solution is in the process of being rolled out within CCLRC and pilot users have been accessing the CCLRC SCARF cluster through the SSO solution using auto-generated low-assurance certificates. Further, through the Diamond-CCLRC SSO committee there are plans for our SSO solution to be used across the Diamond and ISIS facilities. Users can already use the SSO solution as a fast and convenient way of accessing any Grid resource (including the NGS) for which they are authorized, once they have uploaded a real UK e-Science proxy to the SSO MyProxy server.

4 Security Evaluation

From the point of view of the Resource Manager, our SSO solution is an implementation of the Grid Security Infrastructure specification as described by Von Welch *et al* [13]. Trust is established upon presentation by the user of an X.509 identity proxy certificate [14, 2]. As with choosing to trust a classic CA, the Resource Manager must choose to trust the SSO CA or another classic CA to enable SSO access for a user by installing the root certificate of the CA. Once this is done, the user is granted access to the resource once their DN is registered in the normal way.

4.1 SSO MyProxy Server

Our SSO solution requires short-lived X.509 proxy certificates (of real UK eScience certificates) to be stored on a MyProxy server unencrypted so authentication is performed transparently. The MyProxy server thus represents a potential security problem if it is not adequately secured. However, it must be noted that the lifetime of the proxy certificates stored on this server may be set to a short period of time. Ideally this should be no longer than the lifetime of the original authentication credential used to generate the proxy certificate, but this may not be practical or useful. In the case of users with certificates from a classic CA, this is the lifetime of their long-lived X.509 certificate. In the case of users without such a certificate, this lifetime is the lifetime of the user's Kerberos token (generated in the case of CCLRC by Microsoft AD Key Distribution Centre). It should be noted that this is 10 hours by default; as such, using our SSO solution without long-lived X.509 certificates may not be beneficial to users who need to launch Grid jobs that are likely to last longer than 10 hours.

4.2 Communication Channels

All communication between the user, the MyProxy server or MyProxy CA, the site authentication structure and Grid resource provider is performed over secured channels. This is done by making use of either Kerberos tokens belonging to the user or the MyProxy server, or X.509 certificates in the case of communication between the resource provider and the MyProxy server.

Using our SSO solution alleviates the requirement for users to authenticate themselves by entering a password if the user already possesses a Kerberos token, which could be transparently generated during login to the user's computer. This happens automatically with the Windows operating system if the computer is registered on the domain. In the case of Linux, this can be made possible using a solution such as Vintella. If this is not the case, a user can still gain access to the Grid resource provider, albeit not transparently, using our solution by providing their AD login details to the MyProxy CA server, which then retrieves a Kerberos token by calling out to the AD service solely for the purpose of generating a short term proxy certificate.

4.3 Java GSI-SSHTerm

The Java GSI-SSHTerm user interface is distributed both as a trusted applet and as a standalone application. In both cases the program needs read and possibly write access to the local disk. If the user has their certificate stored locally on disk or in a browser, the program needs to read files and libraries locally, which could include the execution of native code. If run for the first time on a computer, then the UK e-Science root certificate and signing policy is installed in the default location onto that computer, if these files have not already been installed. This is to facilitate running other Grid or CoG kit software in the future.

When run as an applet, such disk access can only be performed if the applet has been given explicit permission to do so by the user. This is done by accepting an object signing certificate issued by the UK e-Science CA (the ca-operator certificate). This gives the user the assurance that the program has not been tampered with, e.g. no Trojan has been intro-

duced to gain unauthorized access to the user's certificate. We plan to investigate the use of Java Web-Start technology in the future to give the similar level of assurance to users when running the program as a standalone application.

4.4 Certificate Security

The existence of inadequately protected X.509 certificates have in the past been a cause of security concern. Ideally the user should have a minimal number of certificates necessary to work with the Grid, and have a copy backed up in a secure offline location. Our SSO solution alleviates the need to have long term certificates issued by a classic CA to make use of local Grid facilities. However, if a user needs a long term certificate then the GSI-SSHTerm provides the additional security option of having the user's only certificate in their browser.

Unlike generating a proxy certificate in the traditional way using the MyProxy client tool, proxy certificates generated by the GSI-SSHTerm or retrieved by a MyProxy server are not stored as a file but rather in resident memory. This minimizes the possibility of a proxy certificate being intercepted on the client machine.

We assert that if reasonable effort is spent safeguarding the deployment of this solution and educating users to use it in a responsible manner, our SSO solution using Java GSI-SSHTerm is no less secure than existing Grid access mechanisms.

4.5 Password Security

A SSO infrastructure potentially *improves* the security of the authentication infrastructure:

- If users may remember fewer passwords – ideally, a single one – they are less likely to write them on notes and stick them to their monitor.
- If users are forced to remember more than one password, because systems are not integrated, they are very likely to reuse their passwords. This is particularly bad in the case of the federal (Active Directory or Kerberos) password, because site policy dictates that this password must be stored only in the central service (KDC). If users reuse *this* password, storing it in potentially less secure places, then site security may be compromised.
- The password is validated by a central service, which means that it can be checked for strength, according to site policy. This is not true for the password that protects the private key, or the browser's keystore, but at least those do not, hopefully, leave the user's desktop machine.

The security improvement is likely to be greater for security-novice or non-technical users who are not used to remembering complicated passwords.

5 Future Work

The work in SSO is progressing in a number of directions. As mentioned in Section 3.2 the CoG libraries, which have been modifed to support SASL/Kerberos, can be used in many other contexts. It is key that the vast majority of site Grid resources support single sign-on, in a consistant way, if the wider single sign-on project at CCLRC is to be successful, therefore work is in progress to use this technology in other Grid related user interfaces, for example with the Grid portals work at Daresbury Laboratory.

An additional area of future work is integrating Shibboleth into our SSO framework. JISC will be deploying a Shibboleth infrastructure [15] for UK academia. Meanwhile, we can build on work done in the various Shibboleth projects concurrent with our work, such as the ShibGrid project at Oxford/CCLRC and the Shebangs project at Manchester (see [16] for details of these and other Shibboleth/Grid projects). The ShibGrid project aims to provide Shibboleth access to NGS, and its architecture is similar to that of Figure 1, with Shibboleth cookies instead of Kerberos tokens. Thus, at least in theory, it should be easy to integrate Shibboleth authentication into the terminal. One potential complication is that the current versions of Shibboleth are entirely web-based: they depend on the HTTP protocol and redirects, etc.

6 Acknowledgments

The authors wish to thank the GOSC (Grid Operations Support Centre) for funding this work. They also wish to express their gratitude to all early users within CCLRC, and in particular those of the CCLRC SCARF cluster, who, led by Dr Duncan Tooke, have already put the terminal into production.

This document is typeset with LaTeX2e.

7 Conclusion

The usual definition of Single Sign-on (SSO) is often too narrow to be useful. In this paper, we start with what we see is a more useful definition of SSO based on requirements gathered from users who already use our Grid resources. Working to those definitions, we provide an implementation of a portable terminal to access Grid resources. We have taken an open

source project with GSI extensions, and integrated it with the site infrastructure using MyProxy. Our own software developments and additions to the terminal are of course also released under an open source licence. We have increased the ease-of-installation by deploying it both as a tarball, and as an applet.

We have described the architecture and how it ties the Grid and the site authentication infrastructure together. We have also described how it fits into the larger picture, with a national CA and a Shibboleth infrastructure, and why we see a need for all three (a CA, the Shibboleth infrastructure, and site credential conversion) in the future.

References

[1] David Byard and Jens Jensen. Single sign-on to the grid. In *Proceedings of the 2005 UK e-Science All Hands Meeting*, September 2005.

[2] S. Tuecke, D. Engert, I. Foster, M. Thompson, L. Pearlman, and C. Kesselman. Internet X.509 public key infrastructure proxy certificate profile. Request for Comments (RFC) 3820, June 2004.

[3] Robbie Allen, Joe Richards, and Alistair G. Lowe-Norris. *Active Directory*. O'Reilly, Sebastopol, California, USA, 3rd edition, January 2006.

[4] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter Conference, Dallas, Texas, USA*, pages 191–202, February 1988.

[5] Jim Basney, Marty Humphrey, and Von Welch. The MyProxy online credential repository. *Software: Practice and Experience*, 35(9):801–816, July 2005.

[6] EU Grid Policy Management Authority. Classic authentication policy profile. Available at *http://www.eugridpma.org/igtf/*, Sep 2005. (version 4.03).

[7] Robert Petkus. One-time-password integration at BNL. Available as *http://hepix.caspur.it/spring2006/TALKS/4apr.petkus.otpbnl.pdf*, April 2006. (Talk given at HEPiX conference).

[8] Jason Novotny, Steven Tuecke, and Von Welch. An online credential repository for the Grid: MyProxy. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, USA*, pages 104–114, August 2001.

[9] S. Santesson and R. Housley. Internet X.509 public key infrastructure authority information access certificate revocation list (CRL) extension. Request for Comments (RFC) 4325, December 2005.

[10] Randy Butler and Tony Genovese. Global grid forum certificate policy model. Available at *http://www.ggf.org/documents/GFD.16.pdf*, Jun 2003.

[11] Data Protection Act 1998. Available as *http://www.legislation.hmso.gov.uk/acts/acts1998/19980029.htm*, March 2000.

[12] Gregor von Laszewski, Jarek Gawor, Peter Lane, Nell Rehn, Mike Russell, and Keith Jackson. Features of the Java commodity Grid kit. *Concurrency and Computation: Practice and Experience*, 14:1045–1055, 2002.

[13] Von Welch, Frank Siebenlist, Ian T. Foster, John Bresnahan, Karl Czajkowski, Jarek Gawor, Carl Kesselman, Sam Meder, Laura Pearlman, and Steven Tuecke. Security for Grid services. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12), Seattle, WA, USA*, pages 48–57, 2003.

[14] CCITT recommendation X.509: The directory - authentication framework. CCITT Blue Book, volume VIII, pages 48–81, 1988.

[15] Tom Scavo and Scott Cantor. Shibboleth architecture technical overview. Internet 2 document: draft-mace-shibboleth-tech-overview-02, June 2005. Available at *http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf*.

[16] Von Welch. Report for the GGF 16 BoF for grid developers and deployers leveraging shibboleth. Summary of BoF sessions at GGF 16, February 2006. Available at *http://grid.ncsa.uiuc.edu/papers/GGF16-Shib-BOF-Report.pdf*.