Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy

Tom Barton[1], Jim Basney[2], Tim Freeman[1], Tom Scavo[2],
Frank Siebenlist[1,3], Von Welch[2], Rachana Ananthakrishnan[3],
Bill Baker[2], Monte Goode[4], Kate Keahey[1,3]
[1]University of Chicago
[2]National Center for Supercomputing Applications, University of Illinois
[3]Mathematics and Computer Science Division, Argonne National Laboratory
[4]Lawrence Berkeley National Laboratory

Abstract

This paper describes the recent results of the GridShib and MyProxy projects to integrate the public key infrastructure (PKI) deployed for Grids with different site authentication mechanisms and the Shibboleth identity federation software. The goal is to enable multi-domain PKIs to be built on existing site services in order to reduce the PKI deployment and maintenance costs. An authorization framework in the Globus Toolkit is being developed to allow for credentials from these different sources to be merged and canonicalized for policy evaluation. Successes and lessons learned from these different projects are presented along with future plans.

1 Introduction

The Grid [11] communities have developed an international public key infrastructure (PKI) [18] as well as extensions to standard end entity certificates (EECs) [16] in the form of proxy certificates [40,42]. The combination of this PKI and proxy certificates is used to provide cross-domain authentication, single sign-on, and delegation for a number of large deployments (e.g., [7,31,39]).

As computational Grids have grown, there has been increasing interest in leveraging existing site authentication infrastructure to support this Grid authentication model. For example, Fermi National Accelerator Laboratory has successfully operated an online Kerberos Certification Authority for a number of years to allow its users to leverage existing Kerberos infrastructure for X.509 authentication [38].

In parallel, Shibboleth [37] has been developed by the Internet2 community and is increasingly deployed both in the U.S. and abroad as a mechanism for cross-site access control for web-based resources. Shibboleth utilizes OASIS SAML standards [21,22,29] for authentication and attribute assertions to achieve its purpose.

In this paper we cover recent work by two projects, GridShib [12,43] and MyProxy [1,25], working towards the integration of PKIs with both site authentication infrastructure and Shibboleth in order to achieve large-scale multi-domain PKIs for access control.[1] In section 2 we begin with a brief review of the Globus Toolkit and Shibboleth on which our work builds. In section 3 we summarize our work and lessons learned from the past year. We conclude in section 4 with our plans for the upcoming year.

[1] We stress this infrastructure is for access control and similar point-in-time decisions as opposed to long-term document signing, for example.

2 Prior Work

In this section we provide a brief overview of the Globus Toolkit and Shibboleth on which our work builds.

2.1 Globus Toolkit

The Globus Toolkit [10] provides basic functionality for Grid computing with services for data movement and job submission, and a framework on which higher-level services can be built. Over recent years, the Grid has been adopting Web Services technologies, and this trend is reflected in recent versions of the Globus Toolkit in implementing the Web Services Resource Framework [30] standards. This convergence of Grid and Web Services was part of our motivation for adopting Shibboleth, which is also leveraging Web Services technologies.

The Grid Security Infrastructure [44], on which the Globus Toolkit is based, uses X.509 end entity certificates (EECs) [16] and proxy certificates [42]. In brief, these certificates allow a user to assert a globally unique identifier (i.e., a distinguished name from the X.509 identity certificate). We note that in Grid scenarios there is often an organizational separation between the certificate authorities (CAs), which are the authorities of identity (authentication) and the authorities of attributes (authorization). For example, in the case of the Department of Energy (DOE) SciDAC program [36], a single CA, the DOE Grids CA [6], serves a broad community of users, while the attributes and rights for those users are determined by their individual projects (e.g., National Fusion Grid, Earth Systems Grid, or Particle Physics Data Grid).

Authorization in the Globus Toolkit is by default based on access control lists (ACLs) located at each resource. The ACLs specify the identifiers of the users allowed to access the resource. Also, higher-level services (such as CAS [32]) that provide richer authorization policies exist as optional configurations. As is discussed later, the GridShib project enhances the authorization options of the Globus Toolkit by adding standards-based attribute exchange for both authorization policies and service customization.

2.2 Shibboleth

Shibboleth[37] provides cross-domain single sign-on and attribute-based authorization while preserving user privacy. Developed by Internet2/MACE [19], Shibboleth is based in large part on the OASIS Security Assertion Markup Language (SAML). The SAML 1.1 browser profiles [17,21,34] define two functional components, an Identity Provider and a Service Provider[2]. The Identity Provider (IdP) creates, maintains, and manages user identity, while the Service Provider (SP) controls access to services and resources. An IdP produces and issues SAML assertions to SPs upon request. An SP consumes SAML assertions obtained from IdPs for the purpose of making access control decisions. Shibboleth specifies an optional third component, a “Where Are You From?” (WAYF) service to aid in the process of IdP discovery.

The Shibboleth specification [2] is a direct extension of the SAML 1.1 browser profiles [21]. While the SAML 1.1 browser profiles begin with a request to the IdP, the Shibboleth browser profiles are SP-first and therefore more complex [34].

In addition to the browser profiles, Shibboleth specifies an Attribute Exchange Profile [2]. On the IdP side, a Shibboleth Attribute Authority (AA) produces and issues attribute assertions, while a

[2] For the purposes of discussion, we adopt SAML 2.0 terminology [15] throughout this paper, although our work is currently based on SAML 1.1 technology.

subcomponent of the SP called an Attribute Requester consumes these assertions. Our work builds on Shibboleth attribute exchange with a focus on authorization and access control in the Globus Toolkit.

The current implementation of the specification is Shibboleth 1.3 (released July 2005), which has become our primary development platform. We describe extensions and enhancements to the Shibboleth Identity Provider and Service Provider components later in this paper.

3 Recent Results

In this section we provide a summary of our results from the past year.

3.1 MyProxy

MyProxy began as an online credential repository for X.509 proxy credentials encrypted by user-chosen passphrases [28]. Users authenticate to the MyProxy service to obtain short-lived (per session) proxy credentials that are delegated from credentials stored in the repository. This gives users convenient access to proxy credentials when and where needed, without requiring them to directly manage their long-lived credentials. The latter remain protected in a secure repository, where the repository administrator can monitor and control credential access.

In the past year, we have extended MyProxy to better integrate with existing site infrastructure and to make it easier for users to bootstrap their X.509 security context. New developments, described in the following sections, include management of trust roots, standards-based integration with site authentication, and the ability to act as a Certificate Authority (CA).

3.1.1 Managing Trust Roots

A user's X.509 security context includes an end entity or proxy credential, one or more trusted CA certificates, and certificate revocation information in the form of Certificate Revocation Lists (CRLs) [16] or online certificate status protocol (OCSP) [26] responses. Users can run the MyProxy Logon application to obtain their complete security context from the MyProxy service. The MyProxy administrator maintains a set of trusted CA certificates and configures the server to periodically fetch fresh CRLs. MyProxy Logon fetches the configured CA certificates and CRLs in addition to the user's end entity or proxy certificate and installs them in the local user's environment.

This work is inspired by Gutmann's "Plug-and-Play PKI" [13] which describes a PKI bootstrapping service aimed to make PKI enrollment as easy as adding a computer to the network with DHCP. Gutmann's PKIBoot service can use two methods to bootstrap mutual trust between the un-initialized client and the certificate issuer. The first method uses a shared secret (such as an enrollment password) to generate a Message Authentication Code (MAC) for each message. The second method is a variant of the "baby-duck security model" where the client trusts the first issuer it finds for the one-time bootstrap operation.

A drawback to the shared secret method is it becomes yet another password that users must remember. Common site authentication methods, such as Unix passwords, One-Time Passwords, and Kerberos, allow a service to verify a password entered by the user, but don't allow a service to lookup the user's site authentication password in advance for use in a MAC or other secure password protocol. Thus existing site passwords cannot be used and we must therefore have a unique password for the bootstrap service. In environments where users must bootstrap their PKI context repeatedly as they use different machines, it becomes necessary to maintain a long-lived password or dedicated one-time password stream using S/Key or equivalent.

The baby-duck method is well known to SSH users, who learn the public keys of target hosts in the first connection attempt. This approach is generally accepted as "good enough" given the infrequency of connecting to a target host for the first time and the infrequency of man-in-the-middle attacks in practice relative to keystroke loggers, Trojan horses, viruses, etc.

MyProxy Logon currently supports two approaches to this initial bootstrapping. The first is to use an existing SASL mechanism that supports mutual authentication, such as Kerberos, for the bootstrap operation, leveraging existing site authentication infrastructure. The second is to distribute a trust root for the MyProxy service with the MyProxy client software distribution, recognizing that we trust this software distribution in any case not to capture passwords or otherwise misuse credentials. We have also prototyped the baby-duck approach and are considering it as a lighter-weight alternative.

3.1.2 Site Authentication

The MyProxy service can be configured to allow users to logon with existing site credentials, using Pluggable Authentication Modules (PAM) and/or the Simple Authentication and Security Layer (SASL). Through these mechanisms, users are not required to remember another username and password for the MyProxy service.

Unix/Linux vendors support many PAM modules, including Unix password, One-Time Password, Radius, Kerberos and LDAP. We have successfully tested our MyProxy PAM interface with Radius (and One-Time Passwords), Kerberos and LDAP. PAM also supports access control and monitoring modules to implement standard security policies across multiple services. PAM authentication is based on user interaction, typically through one or more password prompts. In contrast, SASL provides a flexible protocol framework for supporting multiple authentication mechanisms. The primary SASL mechanism used by MyProxy is GSSAPI, which allows users to authenticate with a Kerberos ticket to obtain their X.509 credentials from MyProxy.

3.1.3 MyProxy Certificate Authority

For users that don't already have X.509 credentials to store in the MyProxy repository, the administrator can configure MyProxy to act as an online CA to issue certificates in real time based on site authentication. The administrator must provide a mapping of authenticated usernames to certificate subjects, either in a configuration file or through LDAP. The user authenticates via MyProxy Logon to the MyProxy service, and MyProxy issues a certificate to the user with the subject provided in the mapping file.

By leveraging existing site authentication infrastructure through PAM and SASL, the MyProxy CA provides a lightweight mechanism for sites to distribute X.509 credentials.

3.2 GridShib: X.509 and SAML Integration

GridShib is a software product that allows for interoperability between the Globus Toolkit and Shibboleth. The complete software package consists of two plug-ins: one for the Globus Toolkit (GT) and another for Shibboleth. With both plug-ins installed and configured, a GT Grid Service Provider may securely request user attributes from a Shibboleth Identity Provider. In this section, we briefly describe both software plug-ins and then describe the profile by which they operate in greater depth.

3.2.1 GridShib for Globus Toolkit

GridShib for Globus Toolkit is a plug-in for Globus Toolkit 4.0. Its primary purpose is to obtain attributes about a requesting user from a Shibboleth attribute authority (AA) and make an access control decision based on those attributes. The plug-in implements a policy decision point (PDP) based on attributes obtained from the AA. A policy information point (PIP) does the actual work of requesting attributes. The separation between PIP and PDP allows the plug-in to be used in flexible ways within the toolkit's authorization framework.

3.2.2 GridShib for Shibboleth

GridShib for Shibboleth is a name mapping plug-in for a Shibboleth 1.3 identity provider. Its main purpose is to allow the servicing of attribute queries from Grid SPs based on the user's X.509 Subject distinguished name (DN). The plug-in allows the attribute authority to map the user's DN to a local principal name. Upon receiving an attribute query, the Shibboleth attribute authority uses this plug-in to map the DN and utilizes the resulting principal name to resolve attributes.

The name mapping is a memory-bound collection of name-value pairs. The name (key) is a canonicalized DN that conforms to RFC 2253 [41]. The value is the local principal name. The collection is initialized when the Identity Provider starts up. The current implementation of the name mapping construct is file-based, that is, the mapping entries are read from an ordinary text file. This text file is similar to the grid-mapfile used by Globus Toolkit.

3.2.3 GridShib Profile

The GridShib Profile is an extension of the Shibboleth Attribute Exchange Profile [2]. The primary difference is the use of X.500 distinguished names (DNs) to identify principals.

The GridShib Profile is designed for a standalone attribute requester, that is, an attribute requester that does not participate in a Shibboleth browser profile. Consequently, the Grid SP does not have access to an opaque handle typically issued by the IdP on the front end of the browser profile. In lieu of a handle, the Grid SP uses the DN obtained from the client's proxy certificate.

The primary use case we consider here is a Grid Client that already possesses an X.509 end entity certificate (EEC). As is often the case in grid-based scenarios, the established user uses their EEC to generate a proxy certificate as part of single sign-on. The proxy certificate is subsequently used to authenticate to Grid SPs as part of the act of requesting service.

We therefore make the following assumptions:

- The Grid Client and the Grid Service Provider (SP) each possess an X.509 credential.
- The Grid Client has an account with a Shibboleth Identity Provider (IdP).
- The IdP is able to map the Grid Client's X.509 Subject DN to one and only one user in its security domain.
- The IdP and the Grid SP each have been assigned a globally unique identifier called a *providerId*.
- The Grid SP and the IdP rely on the same metadata format and exchange this metadata out-of-band.

The GridShib protocol flow, depicted in Figure 1, consists of the following four (4) steps.

Step 1 is the beginning of a normal grid request/response cycle. As usual, the Grid

Client authenticates using their X.509 credentials to the Grid service provider. The Grid SP authenticates the request and extracts the client's DN from the credentials.

At step 2, the Grid SP formulates a SAML attribute query whose `NameIdentifier` element is the DN extracted from the client's certificate in step 1. The Grid SP uses its X.509 credential to authenticate to the AA.

At step 3, the IdP, or more specifically the attribute authority component of the IdP, authenticates the attribute request, maps the DN to a local principal name using the plug-in described earlier, retrieves the requested attributes for the user (suitably filtered by normal Shibboleth attribute release policies), formulates an attribute assertion, and sends the assertion to the Grid SP.

Finally, at step 4, the Grid SP parses the attribute assertion, caches the attributes, makes an access control decision, processes the client request (assuming access is granted) and returns a response to the Grid Client.

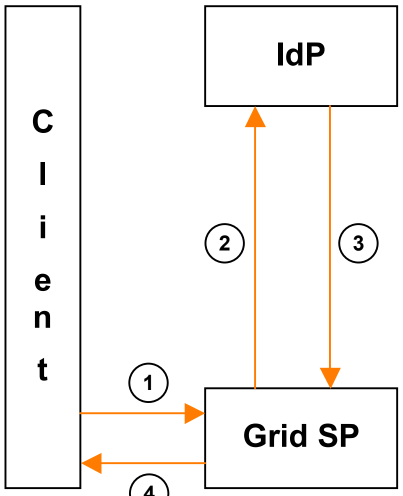

Figure 1 GridShib Protocol Flow

Both the IdP and the Grid SP rely on SAML 2.0 metadata [3,45] for their trust configuration (i.e., the certificates and public keys of the other entity). GridShib for Shibboleth supports a framework for consuming Grid SP metadata whereby the metadata file includes an `EntityDescriptor` element for each Grid SP that the IdP trusts. SAML 2.0 does not define a role for Grid SPs, however, so an extended role of type **AttributeRequesterDescriptorType** has been specified [35] for use with this profile. The defined role of each such entity is basically that of a standalone attribute requester.

3.2.4 GridShib Software

Beta software that implements the GridShib Profile is available for download from the GridShib web site [12]. Source code is available, licensed under the Apache License, Version 2.0.

3.2.5 Current Implementation Limitations

While we believe our current implementation to be sound from a security perspective, the following administrative limitations are recognized:

- The file-based name mapping doesn't scale. The fact that the DN-principal name pairs are read from a file is a major concern. Even if we were to provide administrative tools to manage the name mapping files, the overhead associated with this maintenance would be prohibitive for large user communities. Clearly, this overhead must be eliminated or at least reduced.
- IdP discovery must be generalized. In step 1 of the flow, we assume that a single IdP can assert attributes for all Grid Clients making requests of a Grid Service. A mechanism to allow a mapping between a user and their preferred IdP is needed.
- Metadata production and distribution needs to be automated or simplified.

Trust in a GridShib deployment is based on a bilateral arrangement between IdP and Grid SP. By virtue of the fact that the two entities exchange and consume each other's metadata, a trust relationship is established. The problem is that n entities give rise to $O(n^2)$ bilateral relationships, which does not scale well.

3.3 Globus Toolkit Authorization Framework

As the Globus Toolkit is used by many different projects and by many different Grid communities, it is clear that it cannot mandate the use of particular technologies and mechanisms. Specifically in the area of attributes and authorization policies, the toolkit has to be very flexible to accommodate local preferences regarding assertion formats and usage patterns.

This section enumerates the many certificate and assertion mechanisms that the toolkit has to support. It also describes an attribute collection and authorization framework that deals with the different mechanisms in a consistent manner and that is able to combine authorization decisions from many different sources to yield a single access decision for the invocation request.

3.3.1 Attribute Collection

When a client invokes a request to a service, that service may have to consider many different identity and attribute formats, like X.509 end entity certificates, X.509 attribute certificates, SAML attribute assertions, LDAP attributes, Handle System [14] attributes, and configuration properties.

As it is very common that client requests are made on behalf of other parties, some of the attribute values do not necessarily apply to the requester, but rather to other entities in the delegation chain.

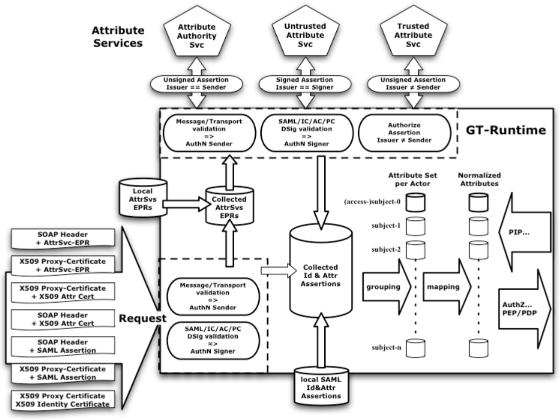
Attribute Services
Attribute Authority Svc
Untrusted Attribute Svc
Trusted Attribute Svc
Unsigned Assertion Issuer == Sender
Signed Assertion Issuer == Signer
Unsigned Assertion Issuer ≠ Sender
Message/Transport validation => AuthN Sender
SAML/IC/AC/PC DSig validation => AuthN Signer
Authorize Assertion Issuer ≠ Sender
GT-Runtime
Local AttrSvs EPRs
Collected AttrSvs EPRs
Attribute Set per Actor
Normalized Attributes
(access-)subject-0
subject-1
subject-2
subject-n
PIP...
SOAP Header + AttrSvc-EPR
X509 Proxy-Certificate + AttrSvc-EPR
X509 Proxy-Certificate + X509 Attr Cert
SOAP Header + X509 Attr Cert
SOAP Header + SAML Assertion
X509 Proxy-Certificate + SAML Assertion
X509 Proxy Certificate X509 Identity Certificate
Request
Collected Id & Attr Assertions
grouping
mapping
AuthZ... PEP/PDP
Message/Transport validation => AuthN Sender
SAML/IC/AC/PC DSig validation => AuthN Signer
local SAML Id&Attr Assertions

Figure 2. Attribute Collection Framework

Furthermore, the attributes can arrive at the service in a number of different ways. Some attributes are "pushed" by the requester, as in VOMS [8] or CAS [32], where the assertion bundle is included with the client request.

Other attributes are "pulled" by the service from attribute services like LDAP, SAML-compatible services like the Shibboleth Attribute Authority, or the Handle System. Note that each of the pull mechanisms uses different protocols.

Lastly, attributes can also be locally stored in (configuration) files on the service side.

The validation of the attribute binding is also dependent on the assertion format and how the information was received. Some attribute bindings are asserted through public key signatures, while others are received unsigned but embedded in protected messages or received over authenticated channels.

Finally, the attribute names and values have to be considered within the context of their definition as well as the context of the issuer. Besides the vocabulary, semantics, and ontology that apply to the attribute bindings, it is also important to understand clearly whether the assertion is only valid in the local context of the issuer or in a global context that requires additional authorization during the validation process.

In order to manage the attribute collection in a consistent manner, the Globus team is in the process of developing a framework depicted in Figure 2. Its purpose is to accept and validate the various attribute assertion formats and mechanisms, to group all the attributes that apply to the same entity together, to translate the names and values into a single format, and finally to make the attribute collections available to the subsequent authorization decision processing phase.

3.3.2 Authorization Mechanisms

As was the case for attribute collection, the processing of the authorization policy enforcement is a similar challenge because of the fact that many formats and mechanisms have to be supported. The applicable authorization policy can come from many different sources, like the resource owner, the resource domain, the requester, the requester's domain, the virtual organization, or intermediaries.

Authorization decisions can be evaluated within the same hosting environment as the policy enforcement point, or can be evaluated by external authorization services. External policy decision points (PDPs), like PERMIS [46], are accessed through the SAML 1.1 authorization query protocol or by using the SAML 2.0 Profile of XACML v2.0 [9].

We have the common delegation-of-rights scenario where one subject can empower others to work on her behalf through the issuing of policy statements. As a consequence, there can be multiple policies and decisions that have to be combined to yield a single decision about the access rights of the requester.

The requester can push some of these policy statements or decisions as authorization assertions, which have to be evaluated by the resource owner. Proxy certificates are simple examples of such authorization assertions. CAS uses SAML authorization decision assertions that are either embedded in proxy certificates or communicated in the SOAP header.

There are many different mechanisms and languages used to express authorization policies, like grid-mapfiles, proxy certificates, SAML authorization decision assertions, CAS policy rules, XACML policy statements, PERMIS policies, and simple ACLs. Note that previously collected identity and attribute values have to be available for the authorization policy evaluations.

3.3.3 Authorization Decision Evaluation

After all the attributes and authorization assertions are collected, and internal and external authorization services are identified, the authorization decision for the access request can be determined.

In order to be able to deal with different authorization mechanisms, the authorization framework uses a PDP abstraction having the same semantics as the one defined in XACML, requiring that each authorization mechanism provides a PDP interface to the framework, each having its own custom decision evaluator that understands the intrinsic semantics of the policy expressions. The PDP abstraction allows the framework to use a common interface to interact with the different mechanism-specific authorization decision evaluators, keeping the mechanism-specific evaluations encapsulated. This common interface is mimicked after the XACML request context interface, which essentially presents the decision request as a collection of attribute values for the subject, resource and action. The PDP's evaluated decision result can have the values of *permit*, *deny* or *not-applicable*. Note that the PDP's decision is associated with either the issuer of the policies that were evaluated or with the identity associated with an (external) authorization service.

For each received authorization assertion and for each authorization service, a mechanism-specific PDP instance is created. As each of those PDP instances is queried through the same interface to evaluate authorization decisions, the mechanism-specific details are all hidden behind the abstraction.

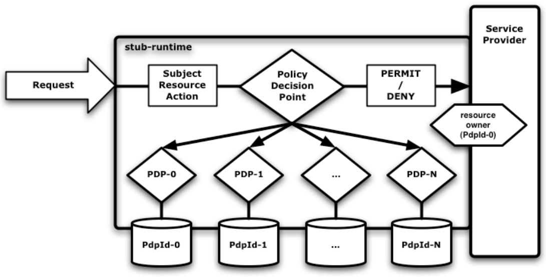

Figure 3. Authorization Framework with PDP Abstraction of Authorization Mechanisms

As shown in Figure 3, a separate *Master PDP* abstraction is used to combine all the different decisions from the various PDP instances in such a way that a single decision reflects the overall evaluated policy. In essence, this Master PDP queries the different PDP instances about the access rights of the requester and potential delegates, and searches for valid delegation decision chains that originate from the resource owner's policy and end with a statement that speaks to the access rights of the requester. The existence of such a valid delegation chain essentially states that the expressed delegation is allowed.

Note that through the use of PDP abstractions, the framework is able to evaluate decisions about delegated access rights for the requester, without the need for explicit support of delegation in the policy languages used in the authorization mechanisms.

3.3.4 Current and Future GT Support

The currently shipping GT 4.0 implementation includes a simplified version of the described attribute collection and authorization framework, but does not fully support attribute-based authorization and has no support for fine-grained delegation of rights. It includes support for proxy certificate delegation, call-out support to SAML 1.1-compliant authorization services, grid-mapfile authorization, and an XACML evaluator.

Enhancements to support Shibboleth and SAML attribute assertions have been added as part of the GridShib effort, and are included in the GridShib beta release.

The full-featured authorization framework is under active development, has produced a number of prototypes, and will ship with our next major release GT 4.2.

4 Next Steps

In this section we discuss our plans for work in the forthcoming year for enabling the seamless integration of Shibboleth/SAML and Grid Security/X.509.

4.1 GridShib

The limitations noted in the previous sections are being addressed. First of all, the file-based name mapping system will be augmented with a database implementation. This will not solve the maintenance problem, but it will make it easier to provide administrative tools. A database implementation will also facilitate the load-balancing of IdPs. (Load-balancing a cluster of IdPs is an ongoing issue in the Shibboleth Project. We do not want to exacerbate this problem.)

One approach to the IdP discovery problem is to include the IdP providerId in the user's X.509 certificate itself. Thus we are planning a modification to MyProxy that produces certificates containing this information. For this to work, we assume initially that MyProxy resides in the same security domain as the IdP. Further work will attempt to relax this restriction.

As mentioned earlier, metadata is an important aspect of GridShib (or any federated identity management system, for that matter). Therefore the following enhancements are being considered:

- provision attribute release policies (ARPs) from Grid SP metadata;
- consume IdP metadata and provision Grid SP configuration; and
- produce SP metadata from the underlying Grid SP configuration.

On the IdP side, tools to produce and consume metadata are being designed. In particular, a tool to automatically produce IdP metadata would be very helpful. (Other projects such as MAMS [24] are working on ARP tools that could take advantage of the attribute requirements called out in SP metadata.) Similar tools for the Grid SP are being developed.

Testing a classic, browser-based Shibboleth deployment remains a challenge. Testing GridShib on top of Shibboleth is even more difficult. To address this problem, we provide a command-line testing tool that tests both a Shibboleth AA and a GridShib AA. A discriminating test strategy is being built around this tool.

To further simplify testing, centralized test services will be deployed. For example, we hope to stand up an on-line GridShib IdP that new Grid SP deployments can leverage for testing purposes.

4.2 Need for Name Binding

In the simplest case, access to a grid service is managed by providing all users with an X.509 end entity certificate (EEC) from a recognized CA, mapping the names in these EECs to another namespace local to the grid service, and using these local names in access control lists. GridShib provides a means of augmenting this approach to identity-based access control with an attribute-based capability: attributes bound to the *distinguished name* in the EEC are marshaled using Shibboleth and filtered through an access control policy to determine access to the grid service.

To broaden the availability of the grid service to more users, additional naming authorities may be recognized. In particular, we wish to enable use of established naming authorities, such as those local to a user's home organization, and authentication tokens other than X.509 EECs. However, we are constrained by the requirement that an EEC must be presented to the grid service, and that only attributes correlated with the distinguished name in that EEC can be marshaled.

This presents two problems. One is the exchange of an original authentication token for a suitable EEC to be presented to the grid service, which is treated elsewhere in this article. The other is mapping the distinguished name in this EEC to the name in the original authentication token, called the *principal name*, so that attributes bound to the principal name can be marshaled by the grid service. Because the principal namespace is not local to the grid service, and to support pseudonymous access scenarios, we propose to collocate this distinguished name to principal name mapping function with the authority for the principal namespace and the attributes that are bound to principal names. This will replace the grid-mapfile associated with the

Shibboleth IdP in the initial GridShib beta product and will also support dynamic binding of principal names to distinguished names in EECs in a manner that enables the Shibboleth AA to map the distinguished name back to its principal name, enabling it to provide attributes for that principal.

4.3 Direct Client-server Use Case

There are two distinct but equally important scenarios in which this name binding must take place. In the first scenario, which we discuss in this section, the client application communicates directly with the service. The second scenario, which we discuss in the next section, involves a web portal intermediary.

When the client application and service communicate directly, end-to-end X.509 authentication is performed as part of the protocol (which is either based on TLS or SOAP with message-level security based on WS-Security [27]). The difficulty in this case is binding the identifier in the user's X.509 credential back to the principal name so that attributes may be obtained.

In this case, we believe that the online CA functionality in MyProxy (described in section 3.1) can be used to solve this problem. As shown in Figure 4, the user obtains short-lived X.509 credentials initially by authenticating to the MyProxy online CA using their principal name and password.[3] The MyProxy CA would then issue the X.509 credential, embedding into it the user's principal name. The service would then extract the principal name and use it when communicating back to the Shibboleth Attribute Authority.

[3] We use "password" here generically to indicate a static or one-time password, Kerberos credential, or any shared secret.

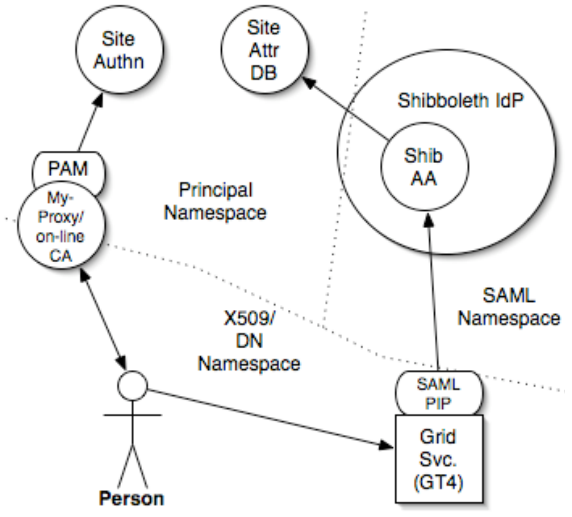

Figure 4: Different namespaces involved in an integrated MyProxy/Grid Service/Shibboleth transaction. The principal name used for authentication (at left) must be transmitted and used for attribute retrieval (upper right).

We note that this approach has a distinct advantage over the current implementation in that the Shibboleth AA does not need to maintain a DN-to-principal name mapping since the principal name is in the SAML query.

One approach is to use CryptoShibHandle [5], a modified Shibboleth handle that encrypts the principal name (along with a nonce and expiration time) into the handle itself. Encryption relies on a symmetric key shared with the Shibboleth Attribute Authority. Used in combination with a non-identifying X.509 DN, CryptoShibHandle preserves privacy by concealing user identity from the Grid service.

An open issue is the appropriate mechanism for embedding the principal name into the X.509 certificate. Current options being considered are to use the Subject Alternate Name or the Subject Information Access extension (sections 4.2.17 and 4.2.2.2 of [16] respectively). One could also embed the principal name into the DN itself (in fact the LionShare security profile [20] specifies precisely this), however we are concerned about placing requirements on the contents of the DN.

We also note that it would be desirable to embed the providerId of the Shibboleth Attribute Authority in the proxy certificate, allowing the Grid service to easily locate the Attribute Authority. This solves the IdP discovery problem discussed earlier

4.4 Portal Use Case

The other use case mentioned in the previous section involves the client using a web browser to access a web server, which in turn accesses Grid services on behalf of the client. This use case is becoming more common as a means to allow for easy access to Grid services with a minimal footprint installation on the client system.

The primary observation in this case is that the portal effectively functions as a "chasm" that must be bridged. Either X.509 or Shibboleth/SAML can be used to authenticate to the portal, but neither has a delegation method that allows for the delegation of authority from the user of a web browser to a portal (see, however, recent work of Cantor [4]). This is the so-called *n*-tier problem ($n > 2$), an active research area.

We note that MyProxy has been used traditionally in the Grid community to enable a portal to use a client's username and password to obtain X.509 credentials for the client. Recent work [23] has also shown that this can be extended to web single sign-on using PubCookie [33]. We believe this approach can be adapted to allow Shibboleth-issued SAML authentication assertions to be used to obtain X.509 credentials from MyProxy[4].

As in the previous section, these X.509 credentials would have the principal name, taken from the `NameIdentifier` element in the SAML assertion, embedded in them. This would allow the Grid service to query the SAML Attribute Authority in an identical manner as described previously.

5 Conclusions

We have presented recent results from the GridShib and MyProxy projects. The goal of both projects is to ease PKI deployment costs by leveraging existing site infrastructure for the establishment of multi-domain PKIs to facilitate policy enforcement.

6 Acknowledgments

The GridShib work is funded by the NSF National Middleware Initiative (NMI awards 0438424 and 0438385). Opinions and recommendations in this paper are those of the authors and do not necessarily reflect the views of NSF.

The MyProxy work was funded by the NSF NMI Grids Center and the NCSA NSF Core awards. The online CA work was implemented at LBNL.

We thank the Internet2 Shibboleth development team for their continued cooperation.

"Globus Toolkit" is a registered trademark of the University of Chicago.

"Shibboleth" is a registered trademark of Internet2.

7 References

1. Basney, J., Humphrey, M., and Welch, V. "The MyProxy Online Credential Repository," Software: Practice and

[4] The newly formed "ShibGrid" projects, ShibGrid and SHEBANGS, sponsored by the UK Joint Information Systems Committee has similar goals and we expect to collaborate on or leverage their work in this area.

Experience, Volume 35, Issue 9, July 2005, pages 801-816.

2. Cantor, S. et al., Shibboleth Architecture: Protocols and Profiles. Internet2-MACE, 10 September 2005. Document ID internet2-mace-shibboleth-arch-protocols-200509 http://shibboleth.internet2.edu/docs/internet2-mace-shibboleth-arch-protocols-latest.pdf
3. Cantor, S. et al., Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, 15 March 2005. Document ID saml-metadata-2.0-os http://www.oasis-open.org/committees/security/
4. Cantor, S. SAML 2.0 Single Sign-On with Constrained Delegation. Working Draft 01, 1 October 2005. Document ID draft-cantor-saml-sso-delegation-01 http://shibboleth.internet2.edu/docs/draft-cantor-saml-sso-delegation-01.pdf
5. CryptoShibHandle https://authdev.it.ohio-state.edu/twiki/bin/view/Shibboleth/CryptoShibHandle
6. DOEGrids Certificate Service, http://www.doegrids.org/
7. Enabling Grids for E-sciencE (EGEE), http://public.eu-egee.org
8. EU DataGrid, VOMS Architecture v1.1. 2003. http://grid-auth.infn.it/docs/VOMS-v1_1.pdf.
9. Anderson, A. and Lockhart, H. SAML 2.0 Profile of XACML v2.0. OASIS Standard, 1 February 2005. Document id: access_control-xacml-2.0-saml-profile-spec-os
10. Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.
11. Foster, I., and Kesselman, C. (eds.). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
12. GridShib: A Policy Controlled Attribute Framework http://gridshib.globus.org/
13. Gutmann, P. Plug-and-play PKI: A PKI your Mother can use. Presentation given at the 12th USENIX Security Symposium, Washington, 2003.
14. The Handle System, http://www.handle.net/, 2005.
15. Hodges, J. et al. Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005.
16. Housley, R., Polk, W., Ford, W., and Solo, D., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, IETF, April 2002.
17. Hughes, J. et al. Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS, May 2004.
18. International Grid Trust Federation, http://www.gridpma.org/, 2005.
19. Internet2 Middleware Architecture Committee for Education (MACE) http://middleware.internet2.edu/MACE/
20. The LionShare Project http://lionshare.its.psu.edu/main/
21. Maler, E. et al., Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS, September 2003.
22. Maler, E. et al., Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS, September 2003.
23. Martin, J., Basney, J., and Humphrey, M. Extending Existing Campus Trust Relationships to the Grid through the Integration of Pubcookie and MyProxy. 2005 International Conference on Computational Science (ICCS 2005), May 22-25, 2005. Emory University, Atlanta, GA, USA.
24. Meta-Access Management System (MAMS) http://web.melcoe.mq.edu.au/projects/MAMS/
25. MyProxy Credential Management Service http://grid.ncsa.uiuc.edu/myproxy/
26. Myers, M. et al. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP). RFC 2560, IETF, 1999.
27. Nadalin, A., et. al., Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), March 2004.
28. Novotny, J., Tuecke, S., and Welch, V.. An Online Credential Repository for the Grid: MyProxy. In Proceedings of the Tenth International Symposium on High Performance Distributed Computing

(HPDC-10). IEEE Computer Society Press, 2001.
29. OASIS Security Services (SAML) TC http://www.oasis-open.org/committees/security/
30. OASIS Web Services Resource Framework (WSRF) TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
31. OpenScienceGrid, http://www.opensciencegrid.org
32. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.
33. Pubcookie: open-source software for intra-institutional web authentication http://www.pubcookie.org/
34. Scavo, T. et al., Shibboleth Architecture: Technical Overview. Internet2-MACE, 8 June 2005.
35. Scavo, T. et al., SAML Metadata Extension for a Standalone Attribute Requester. Committee Draft 01, 11 April 2005.
36. Scientific Discovery through Advanced Computing (SciDAC), http://www.scidac.org, 2001.
37. The Shibboleth Project http://shibboleth.internet2.edu/
38. Skow, D., Use of Kerberos-Issued Certificates at Fermilab. GGF-15 Community Activity: Leveraging Site Infrastructure for Multi-Site Grids. October 3, 2005. http://www.ggf.org/GGF15/presentations/DDS_20051003_kca.ppt
39. TeraGrid Project, http://www.teragrid.org, 2005.
40. Tuecke, S., Welch, V. Engert, D., Pearlman, L., and Thompson, M., Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, RFC 3820, IETF, June 2004.
41. Wahl, M., Kille, S., Howes, T., Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, IETF, December 1997. http://www.ietf.org/rfc/rfc2253.txt
42. Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S. and Siebenlist, F., X.509 Proxy Certificates for Dynamic Delegation. Proceedings of the 3rd Annual PKI R&D Workshop, 2004. http://middleware.internet2.edu/pki04/proceedings/proxy_certs.pdf
43. Welch, V., Barton, T., Keahey, K., Siebenlist, F., Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration, Proceedings of the 4th Annual PKI R&D Workshop, 2005.
44. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S. Security for grid services. In Twelfth International Symposium on High Performance Distributed Computing (HPDC-12). IEEE Computer Society Press, 2003.
45. Whitehead, G. and Cantor, S., Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x, Committee Draft 01, 15 March 2005.
46. D.W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. Future Generation Computer Systems, 19(2):277-289, February 2003.