Shibboleth as a Tool for Authorized Access Control to the Subversion Repository System

Linh Ngo
University of Arkansas
lngo@uark.edu

Amy Apon
University of Arkansas
aapon@uark.edu

Abstract—**Shibboleth is an architecture and protocol for allowing users to authenticate and be authorized to use a remote resource by logging into the identity management system that is maintained at their home institution. With Shibboleth, a federation of institutions can share resources among users and yet allow the administration of both the user access control to resources and the user identity and attribute information to be performed at the hosting or home institution. Subversion is a version control repository system that allows the creation of fine-grained permissions to files and directories. In this project an infrastructure, Shibbolized Subversion, has been created that consists of a Subversion repository with an Apache web interface that is protected by a Shibboleth authentication system. The infrastructure can allow authorized and authenticated data sharing between institutions yet retains simplicity and protects privacy for users. In addition, it also relieves local administrators from the task of having to perform extra account management for users from other institutions. This paper describes the Shibboleth and Subversion systems, the implementation of the file sharing infrastructure, and issues of attribute maintenance, privacy and security.**

Index Terms—**Fine-Grained Access Control, Authentication, Authorization, Shibboleth, Subversion**

I. Introduction

Educational and research activities are not confined to a single institution, but are performed collaboratively among cooperating institutions across the country or even around the world. As a result, there is a need for the development of resource sharing infrastructure between geographically separated institutions under different administrative domains. For example, it is not uncommon for a group of scientists from several institutions to collaborate on a proposal, or for a group of educators, also from several different institutions, to collaborate on the development of course or training materials. The documents that are developed need to be shared among the project participants in an authorized and easy-to-use manner. The focus of this project is to develop a system for sharing documents such as data files, code, research papers, proposal documents, course materials, and others, in an authorized manner within a collaboration group of individuals from two or more institutions.

It is relatively easy to allow several users to have general access to a repository by providing individual accounts to that repository. It is also relatively easy, given that accounts have been set up, to provide fine-grained access for individuals or groups at the directory or file level using standard Unix or database access permissions. However, the administration of the system, including the maintenance of individual user accounts and permissions for various levels of group access, becomes much more complex and difficult if the number of users is increased to several hundred, if these several hundred user accounts are changing continuously, and if the user accounts are spread across several institutions. Even the simplest case typically requires solving a number of non-technical difficulties. For example, suppose that a group of researchers at University A need to access data at University B. In general to allow this access may require a long distance call, working across different time zones with different work load and schedules, and navigating different internal politics. A fine-grained access control method that allows a certain degree of independence for both the resource provider and resource users is needed.

The provision of a system for document sharing must address issues such as user account management, access control of the shared data, and ease of usage. The system must allow a degree of simplicity for both administrators and users, and must have an authorization system flexible enough to allow fine-grained access control at the user and group level.

To address these issues, a shared repository system has been created with the following characteristics:

- The system is a shared repository with open access for trusted institutions.
- The test of authentication of a user's identity is separated from the test of authorization to access with certain privilege any particular file or directory in the repository.
- No additional identity provider is required for the group of cooperating institutions or individuals. Authentication is done by each individual's institution using the login name and password that is provided to the individual by the home institution.
- Authorization is performed by matching user attributes with resource properties. User attributes are administrated at the home institutions along with the user's local institution accounts. Resources properties are maintained by the administrators of the target repository resources.
- The degree of fine-grained access control can be manipulated as needed.
- Basic requirements such as security and authenticity are guaranteed.
- The system also allows a user to access a repository without revealing personally identifying information, if this capability is allowed by the resource administrator.
- The system can be run on different platforms and no extra installation is required on the client side.

This repository system addresses communication and administration issues between the resource provider of the shared repository systems and the administrators in the authenticating institutions. The constructed document sharing system utilizes the existing identity providers from different institutions in order to further understand the difficulties of working in a federated community.

II. BACKGROUND

The system that has been developed is based on Subversion, a well known open source document repository system, and on the Shibboleth open source system for managing federated access to shared resources. This section gives background on version control systems, including Subversion in particular, and Shibboleth. The section also discusses the nature of access control architectures.

A. Version Control Systems

Version control systems have been used historically in the engineering and software development environments to manage the development of source code and other engineering documents associated with the development process. A version control system typically allows a user to "check out" a document for either read or write access. If a document is checked out for write access then, at minimum, other participating members of the group will be alerted to the possible change and can avoid making modifications to the document at the same time. Typical features of a version control system include the ability to check out documents, synchronize different changes from different users to a document, and reverse these changes back to an earlier version of the document.

A number of version control systems are commonly used, including Revision Control System (RCS) [1], Project Revision Control System (PRCS) [2], Concurrent Version System (CVS) [3], and Subversion [4].

Subversion is an open source version control system. Subversion has many features similar to a traditional version control system and overcomes some limitations of traditional version control systems. One of the new features of Subversion is versioned metadata, which plays an important role in the shared repository system developed here. Metadata is information about a file such as file name or access permissions. With versioned metadata, a set of properties can be assigned for each file and directory of the repository in the form of keys and their values. Furthermore, these properties can also be versioned, which means that access permissions can be tracked over time to see which groups or users have historically had access to files and directories. Due to this characteristic, Subversion was chosen to be the repository in this storage system.

B. Shibboleth

Shibboleth is a project of the Middleware Architecture Committee for Education (MACE) [5] and offers a powerful, scalable, and easy-to-use solution for authentication and authorization access control. Shibboleth has been under development since 2001, is a stable tool, and has been incorporated into National Science Foundation's Middleware Initiative (NMI) Release 9 [6]. The Shibboleth system is able to:

- Utilize existing campus identity and access management infrastructures to authenticate individuals and then send information about them to a resource site. The resource provider can set policy and make an authorization decision based on the information that is provided by the campus identity and attribute information systems.
- Support collaborations between campuses, organizations, and off-campus vendor systems.
- Authenticate and authorize based on attributes only. It is possible to allow access without revealing a user's identity, which allows the user's privacy to be protected if this is desired.

Shibboleth consists of three main components: the Identity Provider, the WAYF (Where Are You From) server, and the Service Provider. Also, the system requires the existence of a certificate authority that is trusted among all components. The steps of the Shibboleth protocol are described next, followed by a more detailed discussion of each of the components of the Shibboleth architecture.

B.1. Shibboleth Protocol. The steps of a Shibboleth session are illustrated in Figure 1. The steps are numbered and labeled using the underlying HyperText Transport Protocol (HTTP) commands (e.g., GET, POST) and proceed as follows: First, the user contacts a Target Resource that is protected by Shibboleth (Step 1). In this step the user uses a browser to access a web site that is has been enabled to use Shibboleth for

authentication and authorization. The Target Resource is illustrated in the box labeled "Server Provider".

In the next series of steps, the Service Provider component redirects the user to the WAYF server so that the user can select a local institution with which to authenticate (Steps 2, 3, 4, and 5). The WAYF is configured with the names of all institutions in the virtual organization and also the corresponding Internet address of the Identity Provider of each institution. The user selects his or her home institution and the underlying software redirects the user request to the Identity Provider of the chosen institution (Step 6). The user is prompted to enter a login name and password for the institution (Step 7). After authentication, the Single Sign-on Service (SSO) at the home institution confirms the identity of the user and returns a handle to the Service Provider that identifies the user for the remainder of the session (Steps 8 and 9).

After the user is authenticated, a separate step is performed to determine if the user is authorized to use the requested resource. Using the session handle, the Service Provider requests the required attributes of the user from an Attribute Repository (not shown in the figure). The Attribute Repository may be maintained at the user's home institution, or may be maintained by a virtual organization for a group of resources that are shared within the virtual organization. The request for attributes is shown in Step 10. The release of particular attributes can be allowed or denied based on how the user or the administration has set attribute release policies.

Finally, the Service Provider receives the attributes (Step 11). An Assertion Consumer service component of the Service Provider compares the user's attributes with the resource requirements. If the attributes match the requirements then the user is authorized to use the resource (Step 12).

The identity provider and attribute repository used in this project consist of a single server. In particular, the server is a test LDAP server that mirrors the capabilities of the local campus LDAP server of the University of Arkansas. The test Identity Provider is used to avoid implementing untested attributes into the campus main authentication server. Trusted communication is established to other identity providers, including the local campus LDAP server at the University of Arkansas and the identity provider at the University of Missouri. The tradeoffs in using a single server for both the identity provider and the attribute repository will become more clear in the section on the EduPerson schema. While not using a separate attribute repository reduces many of the technical tasks of administration and configuration, this strategy creates several difficulties in communicating and agreeing about unique attribute settings between the service provider and the identity providers.

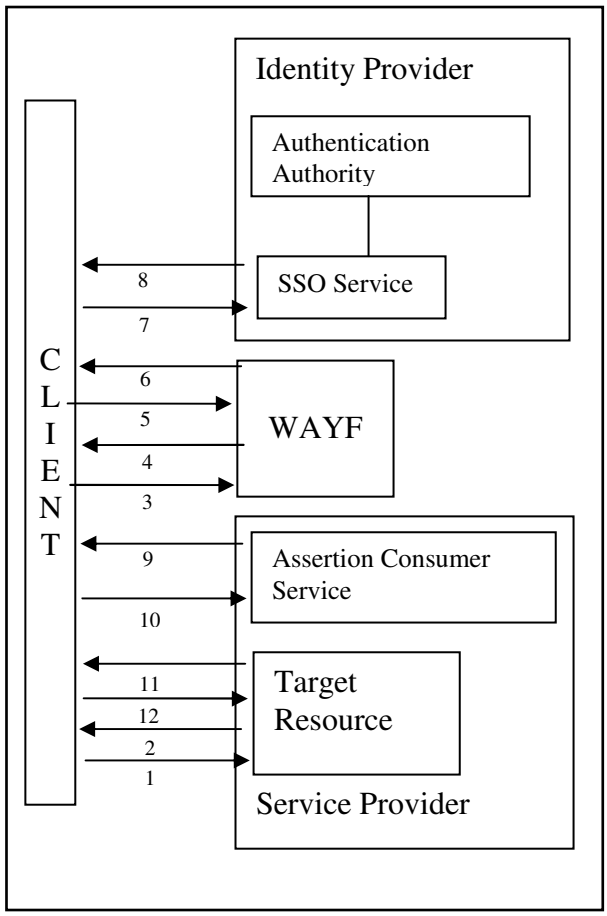

Figure 1: Overview of Shibboleth Architecture [5]

B.2. Server Certification. The Shibboleth protocol depends on the existence of a trust relationship between the various components of the Shibboleth architecture, including each Service Provider and each Identity Provider. In Shibboleth this trust is typically guaranteed through the use of a common Certificate Authority (CA). Each component acquires a certificate that is signed by the common CA The Bossie Certificate Authority created by the University of Wisconsin is used in this project [7].

The Bossie CA provides a very minimal level of trust, but this level of trust is sufficient for the prototype testing for the components in this project. A Bossie certificate was installed at the Identity Provider at the University of Missouri. However, the Identity Provider at the University of Arkansas, which is based on the local campus LDAP server, uses commercial Verisign certificates. With only these installed certificates the Arkansas Identity Provider did not trust the Missouri Service Provider and queries from it failed. This problem was resolved for the prototype testing by manually adding the Bossie server certificate of the Missouri Service Provider into the key store of the local campus LDAP server.

B.3. Shibboleth Service Provider. The Subversion Repository is configured as a Shibboleth Service Provider. When a user contacts the repository, the request is forwarded to the Identity Provider for authentication purposes. After being authenticated, the Service Provider

processes the attributes returned by the Identity Provider for authorization/access control. A Service Provider contains three components: Target Resource, Assertion Consumer Service, and Attribute Requester.

Target Resource: The Target resource is the resource that is protected by Shibboleth. As of the current release, Shibboleth only supports web-based applications. That is, the resource has to be accessible through an Internet browser. However, a bridge connection can be created to map between a web browser and a command line based resource.

Assertion Consumer Service: The Assertion Consumer Service is the counterpart of the Single-Sign On (SSO) Service on the Identity Provider side, except that it is located on the Service Provider side. This service processes the authentication assertion from the Identity Provider's SSO Service. After the authentication between the two sites has been established, it continues with issuing the optional attribute request and then proceeds to authenticate and authorize the users based on the result of this attribute request.

Attribute Requester: The Attribute requester is a SAML based attribute request mechanism that queries the Identity Provider for the attributes needed in order for the user to be authorized and authenticated. Once mutual authentication has been established between the Service Provider and the Identity Provider, this communication can be done with a back-channel attribute exchange. This request is optional depending on the security level of the target resource.

The installations of the Shibboleth Service Provider and its prerequisites are straightforward. However, configuration between the Service Provider and the Identity Provider is complicated and may require several emails and telephone conversations between implementers and administrators among the participating sites. The advantage of Shibboleth is that once the installation is complete and the attributes have been agreed upon, then continued user maintenance and resource configuration can be done independently by local administrators.

B.4. WAYF (Where Are You From). A WAYF server is a server listing the Identity Providers that the Service Provider trusts. After contacting the Service Provider, the user's request is forwarded to a WAYF server. Here, the user must choose an associated Identity Provider. After selecting an Identity Provider, the request is forwarded again to the chosen Identity Provider in order to perform authentication. Two WAYFs are used in this project, including a local WAYF created previously for the WebMPI project [8], and a federated WAYF created by the Shibboleth MACE for the InQueue Federation, a public federation for testing purpose [9].

B.5. Shibboleth Identity Provider. The Identity Provider is the located at the user's local institution. Without revealing to the Service Provider the identity of a user, the Identity Provider will guarantee to the Service Provider that the user is legitimate. Upon request, the Identity Provider forwards a list of user attributes to the Service Provider. These attributes have been previously approved by the users for authorization purposes only. The Service Provider determines, based on these attributes, whether the user is authorized to access selected data in the repository. The Identity Provider of Shibboleth consists of four components: the Authentication Authority, the Attribute Authority, the Single-Sign-On (SSO) Service, and the Artifact Resolution Service.

Authentication Authority: The authentication authority is used to issue authentication statements for the parties participating in the communication process. This component is integrated with the local authentication system and depends on the setup of the local system.

Attribute Authority: The attribute authority processes attribute requests [5]. That is, it receives attribute requests from the Service Provider and processes these requests based on the release permissions given by users. All the requests are in the form of Security Assertion Markup Language (SAML) messages and utilize Secure Socket Layer (SSL)/Transport Layer Security (TLS) or SAML message signatures for mutual authentication [5].

Single-Sign-On (SSO) Service: The SSO Service is the location to which users are directed by the Service Provider. This module performs authentication between the users and their local institutions. After this process, users are directed through a transfer service back to the Service Provider or to an error page depending on the authentication. This service is not a SAML service but an HTTP resource [5].

Artifact Resolution Service: Artifact Resolution Service is a SAML protocol [5] that binds the end-point controlled by the Identity Provider in order to resolve a SAML authentication assertion into corresponding assertions from the requests of the Service Provider.

In this project, the Identity Providers are hosted by the member institutions of the Great Plain Network (GPN) [10].

C. Fine-grained access control

C.1. Access control methods. An access control system consists of an access control policy and an access control mechanism. Normally, these two components both belong to the central administration under the form of an access control list for policy and a mechanism to match users with this list. However, this practice also carries several serious shortcomings:

- Scalability is an issue when the number of users increases.
- There is extra administrative burden in maintaining attributes for users from other institutions.
- Adding and removing users can be slow due to the communication delay between institutions, which can lead to reduced productivity as well as security leaks.
- Privacy of users can be compromised when attributes are released to Resource Providers.

Shibbolized Subversion is based on Attribute Based Access Control (ABAC) [11]. Shibboleth allows the exchange of attributes between its identity provider and target provider, and these attributes are from the user's account on the identity provider side. In this method, the

Subversion directories are marked with specific properties. Only users whose attributes match with these properties can access the directories. While still maintaining the same level of security as traditional access control methods, this method divides the burden of controlling authorization evenly between user side and repository side. Also, this method gives administrators on the repository side the ability to approve or deny specific access by users or by specific types of users to their own resources. Since Subversion allows the creators of the data in the repository to actively modify the properties of these data, access to these data can be controlled by the creators down to the file or directory level. Hence, besides providing fine grain access control, ABAC also encourages an equal participation of both sides, the Resource Provider and the Identity Provider in the access control process.

C.2. EduPerson. For resources that are being shared to a large community, it is also to the benefit of the resource provider to have a set of common attributes that can be easily categorized and distinguished. Among the Shibboleth participants, the most popular attribute scheme is EduPerson, which is the default scheme in Shibboleth's AAP.xml file. It defines a series of fields that are most relevant to the academic environment, and these fields are object class definitions for LDAP servers. Several fields in the EduPerson schema are used for authorization purposes in this project:

eduPersonPrimaryAffiliation: This field provides the name of the identity provider that the user is associated with.

eduPersonScopedAffiliation: This field identifies the role of the user within the identity provider. Such role can be staff, student, or administrators, etc.

eduPersonEntitlement: This field contains the access-control attributes. As described in EduPerson specification, this field accepts attributes with multiple values. Consequently, attributes to describe different levels of access control can be applied.

eduPersonTargetedId: This field contains a unique ID that represents the user, instead of the normal login name. This is to satisfy the requirement of protecting the identity of the user, yet provide means for the service provider to backtrack and report to the identity provider in the case of malicious usage. Usually, this ID can be an encrypted combination of several attributes of the user.

The fields discussed above are the ones recommended by the InQueue [12] and InCommon [13] federations. Depending on the institutions, more fields can be added to further describe the personal attributes of the users. However, the more information is required from the users, the better the security and privacy policy has to be in order to prevent legal complications.

III. Design and Implementation

Shibbolized Subversion has been implemented with three main separate modules: the browser interface, the connection scripts, and the repository. These modules are loosely connected by function calls among themselves, and the infrastructure can change without affecting the whole system as long as the interfaces are kept the same. Figure 2 illustrates the overall structure of the Shibbolized Subversion system.

A. Browser Interface and Security

The browser interface for Shibbolized Subversion is created using Perl CGI and HTML. The main purpose of this module is to provide a simple and easy-to-use interface for users while still retaining most of the important commands of Subversion. There are currently five basic Subversion commands implemented in this interface: check out, add, update, status, and commit.

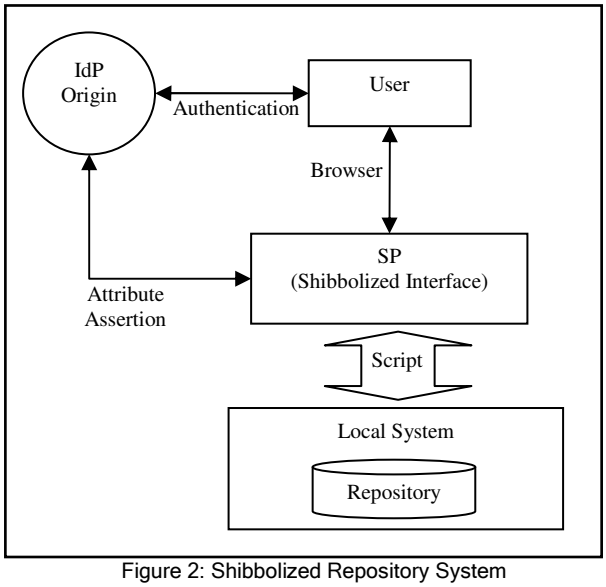

Figure 2: Shibbolized Repository System

A.1. Browser interface structure. The websites of the Shibbolized Subversion user interface are designed using Perl CGI. However, most of the HTML code is embedded in the local scripts called by the CGI programs so that the system can display HTML as well as perform local functions seamlessly. Although these CGI programs carry the initial HTML web page, most of the internal displays of the pages are controlled by the local scripts. Furthermore, the CGI programs have the responsibility of maintaining many default inputs for the local scripts such as name and path of the repository and access control attributes.

A.2. External security. The browser interface performs the function of providing external security of the system. External security provides the access control and authentication for the repository. The primary responsibilities of external security are:

- Authenticate the users with their Identity Provider,
- Provide the users with secured connection for the exchange of password and attributes, and
- Pass the users' attributes to internal security for access control decisions.

These responsibilities are implemented using Shibboleth as an Apache security module for the website. The Shibboleth structure provides identification, authentication, authorization, and accountability [14].

With Shibboleth, identification and authentication are guaranteed by the Identity Provider to make sure that the users are indeed members of the campus organizations and that they are who they say that they are as provided by the users' password. Furthermore, attributes of the users that are passed by the Identity Provider to the Service Provider allow access control decisions to be made. Finally, since these attributes are permitted to be impersonal, the privacy of the users may be protected if that information is not required by the resource.

B. Local Scripts and Internal Access Control

The version of Shibboleth's attribute assertion system used in this project only functions with Web applications [15]. Subversion repository content can be displayed on web pages using Apache's WebDAV. However, only read access can be performed on the web-based Subversion repository. Subversion can only achieve its fullest potential when being accessed with the command line interface. As a result, a mechanism to connect the functionalities of Shibboleth and Subversion is needed. In order to solve this problem, a series of local shell scripts have been used to perform the following functions:

- Receive the attributes passed from Shibboleth
- Perform Subversion commands based on the policies dictated by these attributes
- Display the results of the Subversion commands to a web page

B.1. Attribute passing. The attributes are acquired from the Identity Provider and passed to HTML in the form of HTML headers. For example, the attribute *eduPersonScopedAffiliation* can be accessed by the header of HTTP_SHIB_EP_AFFILIATION. For the prototype system, there are three eduPerson attributes that are requested from the Identity Provider: *ScopedAffiliation*, *Entitlement*, and *TargetedID*. While *ScopedAffiliation* and *TargetedID* are used to help create a unique workspace for the user, *Entitlement* contains all the information concerning the authorization level of the user. After being authenticated, a workspace is created for the user by creating a directory whose name is the concatenation of the values returned for *ScopedAffiliation* and *Entitlement*. From then on, every command and data access related to the user is performed within this directory only. This information is written into a temporary policy file for later use by the local scripts. A system call from the CGI program passes the values of *ScopedAffiliation* and *Entitlement*, and the directory to be checked, out to the scripts.

B.2. Performance of Subversion commands. In processing a Subversion command from the users, the local scripts go through three steps: 1) check out the directory, 2) match user attributes with directory's properties, and 3) process the Subversion command.

All of these Subversion commands require an existing checked out version of the data. Therefore, a "svn checkout" call is needed initially. Immediately after this call, although the data files and directories are now available, the user has no knowledge of the data. One of the limitations of this method is that, if the users do not specify a single directory, the check out script will check out a complete repository database. This will affect the speed of the attributes matching process and take up a larger than normal amount of disk space. The first disadvantage of this method can be reduced by having a large server disk (the checking out process is done completely on the server side). The second disadvantage can be limited by allowing the user to delete the extra data after the copy of the needed document to the local machines is finished. After the data is checked out initially, the authorization process with attribute matching is started.

B.3. Attribute matching. In Shibbolized Subversion, the attribute matching process is divided into two steps: match-attribute and authorize-attribute. Also, in order to simplify the matching process, the following assumptions are made:

- If a user has read access to a folder, he automatically has read access to all the recursive folders and files within that main folder.
- If a user has write access to a folder, it does not mean that he has read access to that folder. In short, read and write access capabilities are two different attributes with equal importance and are granted independently from each other.
- Read access is checked on files and folders, while write access is only checked on folders.

Using a system of hooks implemented within Subversion, a repository's files and folders can each be attached with multiple attributes. During the matching process, these attributes will be recorded in a temporary policy control file to determine read/write access.

When a user's attributes are passed to the script, they are first compared against the attributes attached to the checked out directory. If a match is found, the appropriate HTML code is generated to grant the user access right to the directory. After the attribute matching is completed, the scripts process the appropriate Subversion command based on the choice of the user.

B.4. Display of results. The commands and parameters for Subversion are embedded in the information that the browser transfers to the local scripts. At this step, the local scripts call the Subversion command and return the result to the browser. Here, HTML tags are embedded within the script itself in order to display the contents of the result on a browser.

C. Repository

The repository is designed as a local repository using the Subversion repository system. This is also where the local properties are set up. Depending on the level of security, the owner of the repository can assign properties along different directory tree levels down to the lowest level, the file level. The checked out files are placed in a directory whose name is created as combination of the user's *eduPersonTargetedID* and *eduPersonAffiliation*. This allows a unique storage space for each individual user in the system.

IV. DISCUSSION

The Shibbolized Subversion system satisfies the goals set out for the project. The use of Shibboleth as the authentication service provides:

- Shared repository with access open for trusted institutions. The repository has been shared with the University of Missouri and the local campus directory.
- Authentication done by each individual's institution and no extra login name or password needed.

In addition, using the combination of Shibboleth attributes transfer and Subversion's repository properties, we also satisfy:

- Authorization is done by a series of attributes and matching properties set on the users' local institution accounts and directories in target repositories, respectively
- The degree of fine-grained access control can be manipulated as needed.
- Basic requirements such as security, authenticity, and privacy are guaranteed.

Also, the system can be run on different operating systems with web browsers, and no extra installation is required on the client side.

A. Security

The security of a Shibbolized system depends heavily on the level of the trust relationship between the Identity Provider and the Service Provider. This trust is guaranteed by the SAML protocols and the certificates assigned to the participants by a common trusted CA. If the participants are using different CAs, then all the CAs have to be trusted by all parties. In this setting the compromise of a single CA will lead to the compromise of the whole system.

In production federation, the maintenance of a CA is very strict. For example, the InCommon Federation has a legal contract that requires participant to maintain certain security practices such as separation of the machine containing the CA from the public network and single authority. As a result, the process of getting a certificate can be long and troublesome.

For testing and experimenting purposes, the Bossie certificate allows participants to quickly acquire the certificates. However, since the keypass to acquire a Bossie certificate is publicly broadcast online, it is not a secure method to protect the IdP and Service Provider servers. A production implementation of Shibbolized Subversion would have to address this problem by requiring that a CA with a high level of security be used by all participants in the federation.

B. Privacy

Release of personal attributes is no simple matter. It touches complicated issues related to personal privacy, and it also raises many who-what-when-why-how questions about campus security. These issues can be summarized as [16]:

- Concern from participating institution's compliance and audit offices regarding security and privacy of identity data hosted remotely (UT)
- Demonstrated experience dealing with system-wide projects containing sensitive and non-sensitive information
- Completed security questionnaire detailing security policies and procedures in place
- Required Provider-campus staff to sign security policy authorization of client campus (CSU).

C. Technology

The installation of the Shibboleth service provider is straightforward. However, there are several challenges to setting up the communication between the identity provider and the service provider from different institutions.

The first challenge comes from the differences in the infrastructure between the two institutions. As described in section B.2, even within the local campus infrastructure, the certificates may not match.

Another challenge also arises from the lack of campus attribute infrastructure. Some institutions just do not have the required security infrastructure that is LDAP-compatible, and it is difficult for them to upgrade their facilities to one.

It is difficult for institutions with incompatible infrastructures to overcome administrative difficulties in seeking approval for a new infrastructure. Even in institutions with infrastructure that supports Shibboleth's attribute release scheme, it is a struggle to have the infrastructure set up correctly without interfering with existing regulations. Often, it is the story of "the chicken and the egg," where the institution requires the users to really "want" to use the Shibboleth system before the infrastructure is changed, while the users desire to see the Shibboleth system in action first before they "want" to use it.

V. RELATED WORK

Grid computing is fast becoming a useful technology for large scale research collaborations. For example, the Open Science Grid (OSG) [17] has more 50 participating institutions from inside and outside of the United States. In order to provide adequate access control, the Open Science Grid package uses the Virtual Organization Membership Service (VOMS) [18] and the Grid User Management System (GUMS) [19] for authentication and authorization.

VOMS is part of the European project Enabling Grid for E-SciencE (EGEE). GUMS is developed by the Brookhaven National Laboratory. Figure 3 describes the working relationship between VOMS and GUMS in the OSG software stack. In this procedure, the user first requests a proxy certificate from the VOMS server (Step 1). After authentication, the VOMS server returns a proxy certificate containing the encrypted information of the user (Steps 2 and 3). Next, the user contacts the Job Execution Site and sends the recently acquired proxy certificate (Step 4). The GUMS server decodes this certificate and performs the authorization step (Steps 5 and 6). If the user is authorized, a local ID associate with

the user is returned to the Gatekeeper to start executing the user's job under this ID (Steps 7 and 8).

The relationship between VOMS and GUMS can be compared and contrasted to the relationship between the Identity Provider and the Service Provider of Shibboleth. For example, VOMS holds user identity information. In order to use the grid, the user authenticates to the VOMS server by providing a userid and password that has been previously registered with the server. However, unlike the authentication that takes place with the Shibboleth Identity Provider, this userid is typically unrelated to any userid that may be maintained at the user's home institution and the password is kept separately on the VOMS server. GUMS is a server that matches user proxy information to an access control list associated with a particular service, in a manner that is similar to the Assertion Consumer Service component of the Shibboleth Service Provider.. However, communication between the Shibboleth Identity Provider and Service Provider is in the form of Security Assertion Markup Language (SAML) assertions [5], while the communication from VOMS to GUMS is the exchange of a user proxy certificate [20]. Furthermore, while Shibboleth focuses primarily on user's attributes, VOMS and GUMS usually use user assigned roles for access control [20], which may not provide the same level of fine-grained access control that is available in the Shibboleth architecture.

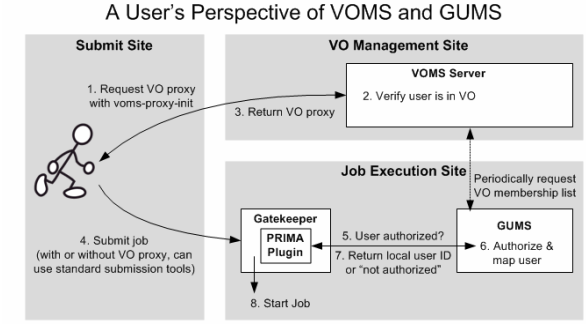

Figure 3: How VOMS and GUMS work together [21]

VI. FUTURE WORK

Although the system specified here works well in a testing environment, there are some limitations and a number of challenges and opportunities for improvement and future work.

Currently, the attributes being used for matching are placed in the field *eduPersonEntitlement*, as this field allows multiple values. However, there is a limit within the EduPerson scheme definition on the length in characters of the *eduPersonEntitlement* field. It is clear that the dependence upon a single field for attribute storage does not scale as the number of resources and attributes in the federation increases. In order to avoid this problem, there are several possible approaches. If the approach using the single identity and attribute server using the eduPerson schema is maintained, then it is possible to either implement new eduPerson fields instead of putting all the attributes into *eduPersonEntitlement*, or to encode attributes in order to reduce the length yet still maintain the versatility of the attributes. Another, perhaps more scalable, alternative is to implement a separate attribute repository, perhaps at the federation level [21].

An additional limitation of the current implementation is the location of ownership permissions of the directories. With the current implementation, the administrator of the Shibboleth Service Provider is also the owner of the repository. In a production environemnt, the directories of a repository may be owned by different people, and each of them would want to have a more active control on his or her data. It is necessary to provide an implementation in which the directory owners can set access permissions to users and groups on their own directories and files without any action on the part of the Service Provider administrator. In additional to giving control to the owners, an implementation of this type would free the administrators from some of the mundane tasks such as setting up properties for the directories. This can be done by implementing the commands *svn propset* of Subversion and making the commands available only for the owner of the repository.

The current design of Shibbolized Subversion allows a convenient and quick access to a small shared data repository. Anytime a user wants to check out a file or set of files, a copy of that data is created on the Subversion server. This technique will not scale to very large data repositories. The problem can be alleviated by transferring the checked out data to the user's local site. However, a stub of the checked out directory still needs to remain at the repository. The stub can be used to guarantee that the Subversion hooks are in place so that the check in process can be done later.

Currently, this system is set up for users to personally access data. However, it is possible in the future to further enhance the system of trust so that we can not only trust people but also other services. For example, user A wants to use the WebMPI service located at institution B to process the data located at institution C. This model would require a more complicated trust relationship between the institutions and would lead to more cooperation opportunities.

The source code for this project is available at http://archie.csce.uark.edu/gpn/ [22] [23].

ACKNOWLEDGMENT

This work has been supported in part by Grant #0410966 from the National Science Foundation. This work has also been supported by the Great Plains Network Consortium Middleware Project through the Extending the Reach (ETR) project. ETR funding is provided on behalf of the NMI-EDIT consortium of Internet2, Educause, and SURA, and with support from several statewide university systems and regional networks.

REFERENCES

[1] Revision Control System, http://www.gnu.org/software/rcs/rcs.html

[2] Project Revision Control System, http://prcs.sourceforge.net/

[3] Concurrent Versions System, http://www.nongnu.org/cvs/

[4] Collins-Sussman, B., Fritzpatrick, B., and Pilato, M. "Version Control with Subversion." O'Reilly 2005. http://svnbook.red-bean.com/.

[5] Scavo, T., S. Cantor. Shibboleth Architecture Technical Overview. http://shibboleth.internet2.edu/shibboleth-documents.html

[6] Enterprise and Desktop Integration Technologies (EDIT) Consortium (2006), http://www.nmi-edit.org/index.cfm

[7] Bossie Certificate Server. http://bossie.doit.wisc.edu/cert/i2server

[8] Landrus, K. "WebMPI – A Secure Cluster Web Interface Using Shibboleth" Master's thesis, University of Arkansas, Fayetteville, Arkansas, May 2005.

[9] InQueue's Testbed. http://inqueue.internet2.edu/test.html

[10] Great Plains Network Consortium (2006), http://www.greatplains.net/

[11] Yuan, E., J. Tong, "Attribute based access control (ABAC) for Web services," *Web Services, 2005. ICWS 2005 Proceedings. 2005 IEEE International Conference on Web Services*, no. 569, pp. 11-15 July 2005.

[12] InQueue Home (2006), http://inqueue.internet2.edu/

[13] InCommon Federation (2006), http://www.incommonfederation.org/.

[14] Bruhn, M. Gettes, M. West, A. "Identity and Access Management and Security in Higher Education." *EduCause Quarterly*. No 4. pp. 14-16. 2003.

[15] Welch, Von, Tom Barton, Kate Keahey, Frank Siebenlist. "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration." *Proceedings of the 4th Annual PKI R&D Workshop, 2005.*

[16] Apon, Amy, David Bantz, Mark Crase, Greg Monaco, Miguel Soldi, Ann West. "From Bright Idea to Actual Implementation: A Lifecycle Perspective to Build Trust in Core Middleware Services." *Internet2 Member Meeting* Spring 2005.

[17] Open Science Grid, http://www.opensciencegrid.org

[18] Virtual Organization Membership Service, http://edg-p2.web.cern.ch/edg-wp2/security/voms/

[19] Grid User Managenent System, http://grid.racf.bnl.gov/GUMS/

[20] An Introduction to Privilege (Authorization) in OSG, https://twiki.grid.iu.edu/twiki/bin/view/Integration/PrivilegeOSG

[21] Ionut O. Ciordas, Fine-Grained Authorization in the Great Plains Network Virtual Organization, MS Thesis, University of Missouri-Columbia, August 2007.

[22] Ngo, L. "Shibbolized Subversion." Master's thesis, University of Arkansas, Fayetteville, Arkansas, December 2005.

[23] Great Plains Network Extending the Reach Project Homepage. http://archie.csce.uark.edu/gpn/

Linh Ngo is currently a Ph.D. student at the Department of Computer Science Computer Engineering of the University of Arkansas at Fayetteville.

Amy Apon received a Ph.D. in Computer Science at Vanderbilt University and is currently a Professor at the University of Arkansas. Dr. Apon was the PI on the Extending the Reach (ETR) Project funded to the Great Plains Network (GPN) Consortium by Educause on behalf of the NMI-EDIT Consortium of Internet2, Educause, and SURA, with the goal of building regional middleware infrastructure across the GPN region. Dr. Apon is also the PI on an MRI grant from the National Science Foundation that funds the supercomputing resource at the University of Arkansas, and grants and funded projects from other agency and industrial sources. Dr. Apon is working on a collaborative project funded by NSF with Dr. Thomas Sterling of Louisiana State University to teach High Performance Computing, and teaches other courses at the University of Arkansas. Her research interests include cluster and grid computing systems, performance evaluation of distributed and parallel systems, and the architecture of middleware systems.