Walden: A Scalable Solution for Grid Account Management

Beth A. Kirschner*, Thomas J. Hacker*, William A. Adamson† , Brian D. Athey§
*Michigan Grid Research and Infrastructure Development (MGRID)
{bkirschn,hacker}@umich.edu
†Center for Information Technology (CITI) Integration
andros@umich.edu
§Michigan Center for Biological Information (MCBI)
bleu@umich.edu

University of Michigan, Ann Arbor, Michigan, 48109

Abstract— **A large and diverse consortium of grid clusters, as can be found in a university setting, requires a flexible authorization model that is scalable, extensible and easy to administer. Current approaches to grid authorization suffer from administrative interfaces that don't scale, authorization models that don't provide needed functionality, or both. This paper proposes a solution with initial implementation that uses existing standards to support the requirements of such a consortium of grid clusters. Our solution eliminates the need to manage authentication and authorization on a per-host basis, and implements a mechanism to *temporarily* bind a grid user to a local guest account on grid resource.**

I. Introduction

The Michigan Grid Research and Infrastructure Development (MGRID) project at the University of Michigan is working to establish a computing fabric that binds together existing high performance and desktop scale computing resources on campus. One significant challenge MGRID is facing is mediating access between a community of 175,000 users and resources that are spread across disparate administrative and geographical domains. To accomplish this, MGRID has developed an approach to managing authentication and authorization that integrates Globus [1] with existing campus authentication and authorization mechanisms [2]. Our new approach eliminates the need to manage user identities on hosts that participate in a grid computing environment. We accomplish this by moving user authentication to the client and replacing the static mapping between X.509 identities (Distinguished Names) and local user names in the Globus *grid-mapfile* with a dynamic approach based on kx509 [3] [4] and secure LDAP. With this new approach, which we call $Walden$, we can easily manage authentication and authorization for any number of users from the campus community without managing user identities on individual hosts.

A. Globus Background

The Globus Toolkit [1] provides a secure framework for submitting and executing grid applications on grid clusters. The Grid Security Infrastructure (GSI) [5] provides X.509 [6] credential based authentication and an authorization model based on static mapping of a user's Distinguished Name (DN) to a local userid using a *grid-mapfile*.

While this approach provides the benefits of secure single-signon authentication, it lacks scalability and extensibility in authorization. This is a critical issue for MGRID, since we must be able to provide access for a community of over 175,000 users. Fortunately, Globus 2.4 provides an option to replace the static *grid-mapfile* with a site specific authorization package [7].

B. Our Solution: $Walden$

To meet our goals for MGRID, we required a secure authorization model that could provide:

1) Scalability – Support for several thousand users, both at the authorization level and at the administrative level. We felt that group or role-based authorization would scale better than authorization based on individual identities.
2) Security – Resource administrators must be confident that users have been properly authenticated and authorized. Users must also be unable to repudiate transactions and resource use.
3) Extensibility – Support for detailed authorization rules that restrict access by queues and/or jobmanagers, time of day, software/resource limitations, in addition to simple access control lists.
4) Resource ownership – Resource owners require an administration mechanism to manage access to resources under their stewardship.
5) Guest/Template User Accounts – Resource administrators should not bear the burden of maintaining thousands of local user accounts. The authorization package must support a one-to-one temporary binding between an authorized user and a generic local user id, when no local user account is present [8].
6) Virtual Organizations – Support for fluid communities of users that dynamically create and disband virtual organizations across campus.

We designed and developed *Walden* to meet these needs. *Walden* is valuable and unique for several reasons:

- *Walden* completely leverages the existing University of Michigan institutional user identity and authentication infrastructure. The University thoroughly vets the physical credentials of a member of the campus community before granting a unique campus-wide secure identity [2] to the user. Since the institution has verified the physical identity of the user, we can be reasonably certain that we know the physical identity and attributes of the user presenting credentials.
- *Walden* eliminates the need to manage authentication and authorization on a per-resource basis. Management is moved to a centralized LDAP server, which allows resource administrators to easily manage access for a collection of resources. This feature also simplifies the addition and removal of resources from the campus Grid.
- By permitting the temporary binding of users to local guest accounts, *Walden* allows resource administrators to control the amount of resources they dedicate to a large pool of users by limiting the number of grid guest accounts. This also eliminates the need to manage access on an individual host basis. Note that *Walden* still permits the permanent binding of a user to a local host account.

MGRID is using *Walden* as a middleware component to provide authentication and authorization services for a generic campus grid portal based on the Open Grid Computing Environment (OGCE) [9] and the Comprehensive Collaborative Framework (CHEF) project [10].

The next section of this paper reviews current work in the area of grid authorization. Following that, we describe the implementation of *Walden*.

II. Current Work

This section describes some current work in the area of grid authorization and provides a brief introduction to XACML for the benefit of the reader.

A. XACML Background

The Extensible Access Control Markup Language (XACML) is a language based on the Extensible Markup Language (XML) [11], [12]. XACML is designed to express both access control policies and a request/response language for querying those policies. The goal of XACML is to implement a common access policy langauge across a wide range of services and vendors. The XACML standard is managed by the Organization for the Advancement of Structure Information Standards (OASIS) [11], which is non-profit consortium supported by a large number of information technology vendors.

XACML access control polices express ”who can do what when”, essentially describing resources, users, and when users have permission to access the resources. The request/response language defines the XML format of policy queries and responses.

```
<Rule RuleId="AllowAllGridruns" Effect="Permit">

 <Target>
 <Subjects>
  <!-- defined in policy target -->
  <AnySubject/>
 </Subjects>

  <Resources>
   <!-- defined in policy target -->
   <AnyResource/>
  </Resources>

  <Actions>
   <!-- define in policy target -->
    <AnyAction/>
  </Actions>
 </Target>
</Rule>

<Rule RuleId="DenyAllOthers" Effect="Deny"/>

 <Obligations>

  <Obligation ObligationId=
   "umich:mgrid:obl:GuestUserId"
    FulfillOn="Permit">

   <AttributeAssignment
    AttributeId="umich:mgrid:attrid:UserIdBase"
    DataType="http://www.w3.org/2001/
     XMLSchema#string">
     guest
   </AttributeAssignment>

   <AttributeAssignment
    AttributeId="umich:mgrid:attrid:UserIdCount"
    DataType="http://www.w3.org/2001/
     XMLSchema#integer">10
   </AttributeAssignment>

   <AttributeAssignment
    AttributeId="umich:mgrid:attrid:UserIdFormat"
    DataType="http://www.w3.org/2001/
          XMLSchema#string">
          00</AttributeAssignment>
   </Obligation>
 </Obligations>
```

Fig. 1. Sample XACML Policy

Two agents are involved in access queries: the Policy Enforcement Point (PEP), which is the agent protecting a resource that receives access requests from users; and the Policy Decision Point (PDP), which grants or denies access to the resources based on policies stored in XACML. Typically, a user sends an access request to a PEP, which in turn consults the PDP for an access decision.

Figure 1 shows a fragment of an MGRID XACML policy file. In this fragment, there are two Rules and one Obligation element. The policy described in this fragment allows grid administrators to restrict access to resources and services (e.g. Globus gatekeeper *jobmanagers*) available on resources. Once a policy is well-defined, it shouldn't require much maintenance. A well-defined policy restricts access based on defined user groups, which allows the administrator to add or remove users from a user group referenced in the policy. Membership in an authorized LDAP group permits access to grid resources and services (such as specific Globus *jobmanagers*) that map to certain scheduler queues, software licenses, or even priority-based scheduling.

The first rule (*AllowAllGridRuns*) permits access to any Subject/Resource/Action tuple that matches the attributes specified in the policy target (not shown). *Subjects* defines the user and user group, *Resources* defines the gatekeeper, and *Actions* defines the Globus *jobmanager*. This rule could define further conditions above and beyond what is specified in the policy target. For example, users who are members of group A might be given access to all *jobmanager* actions, while users who are members of group B might only have access to a subset of *jobmanager* actions.

The second rule (*DenyAllOthers*) denies access to any user not matching the acceptance criteria in any previous rule. This XACML policy uses a *Rule Combining Algorithm* where rules evaluating to *permit* override rules evaluating to *deny*. Other rules override algorithms can be configured, such as *Deny-overrides* and *First-applicable*.

The *Obligations* element defines constraints that the PEP (Policy Enforcement Point) is obliged to perform. In this case, the PEP is obliged to create a guest user id formatted according to the given Obligation attributes. Three attributes define the guest username base string, the range of guest usernames, and the formatting of guest usernames, which can be either numeric or alphabetic. This obligation will concatenate a base of ”guest” to an integer between 1-10, whose numeric format should consist of two digits, yielding a name such as ”guest01”.

Managing the information contained in XACML policy files is currently done using text editors. We are working on designing an XACML editor utility to provide a clear and simple user interface. Our goal is to provide a powerful portlet that will allow resources owners to define and manage authorization policies for resources they provide to the MGRID community.

B. Current Work in Grid Authorization

There are many existing and developing authorization solutions available for grid computing. Several of these solutions make use of X.509 Attribute Certificates to bind a user to specified privileges. Examples of these include PERMIS [13], which binds a user to a group with specified privileges, and PRIMA [14], which directly binds a user to specified privileges. These models work well with small user communities, but are not intended for use for a large user community, since the administrative overhead does not scale well. Other authorization models contain only the framework for passing authorization requests, and lack a built-in policy engine. Examples of these include Shibboleth [15] and VOMS [16]. CAS (Community Authorization Server) was designed to work with the Globus infrastructure, but requires resource owners to delegate the administrative rights to the community administrator. MyProxy [17] generates short-term proxy credentials for users from a central server. MyProxy requires the transmission of a login name and password between the user's web browser and a web server, which is undesirable.

Maintaining a large number of user accounts on a large number of hosts is an important issue for grid computing [8]. Globus requires *a priori* provisioning of a local host account for every user in the *grid-mapfile*. Other approaches (such as Condor [18]) involve a many-to-one mapping of user accounts. This mapping makes it difficult to manage a set of users who may interfere with each other on a system (e.g. temporary disk space and shared memory). GridBank [19] proposes a scheme to maintain a pool of template accounts, which are mapped by dynamically adding an entry in the globus *grid-mapfile* when a job is started, and removing the *grid-mapfile* entry when the job is completed. This approach benefits from having a one-to-one mapping between a user and local account for the duration of the job, but is limited by the authorization model of the Globus *grid-mapfile*. The mapping is not persistent, nor is the mapping retained for frequent users. This leads to significant overhead for creating, managing, and tearing down system accounts and account bindings for frequent users.

Globus 2.4 introduced a new authorization callout option (Globus Authorization and Mapping Callout) [7] that extends the *grid-mapfile* lookup with an alternative authorization call. This feature allows a site to replace the Globus static *grid-mapfile* approach with its own authorization models, and was originally developed for the Fermilab Site Authorization Service (SAZ) [20] project. The Fermilab project makes use of the callout feature to retrieve authorization from the SAZ database. It builds on the framework provided by VOMS and PRIMA, but currently supports only a limited set of authorization rules. This approach replaces the *grid-mapfile* with a database, which limits authorization extensibility and flexibility, while the Walden approach replaces the *grid-mapfile* with an XACML policy engine front end and LDAP server back end.

The *Walden* authorization model described in this paper makes use of the Globus callout to replace the static *grid-mapfile* with an XACML policy engine that implements MGRID's access policies. The implementation of *Walden* draws from some ideas described by NASA's Cardea system [21], but adds a scheme for mapping users to local template accounts on each grid cluster. Cardea is an XACML-based authorization model which incorporates greater flexibility in distributing the PEP, PDP and Attribute Authorities, using Web Services (SAML/SOAP) for messaging between components. The PEP and PDP are not tightly coupled to each other, but instead are discovered at runtime. Users are identified by X.509 proxy certificates, but the information necessary to conclude an authorization decision is retrieved by the PDP (Policy Decision Point). Walden is also similar to PRIMA, which uses XACML for coarse-grained decisions and ties the fine-grained decisions to the local user identity, in the form of file access permissions, user quotes, or network access. Walden uses XACML for all privilege decisions (coarse or fine grained).

Walden uses an XACML-based authorization model, with the typical PEP, PDP, and Attribute Authoritites. Although Walden does not employ web services to distribute the PEP and PDP across different nodes, it does add the capability of dynamically binding users to guest local userids. Like Cardea, users are identified by X.509 proxy certificates, and the information necessary to conclude an authorization decision is retrieved by the PDP.

The next section describes the implementation details of $Walden$, and the $Walden$ authentication and authorization protocol.

III. Implementation

There are three major sequential components in our implementation of Walden: (1) user authentication, which occurs on the user's workstation; (2) establishing the user's local resource identity and checking authorization, which is the main focus of Walden; and (3) managing the local identity in an efficient and scalable manner. This section reviews the issues for each of these steps, and describes in detail how our approach provides a scalable and secure solution for each issue.

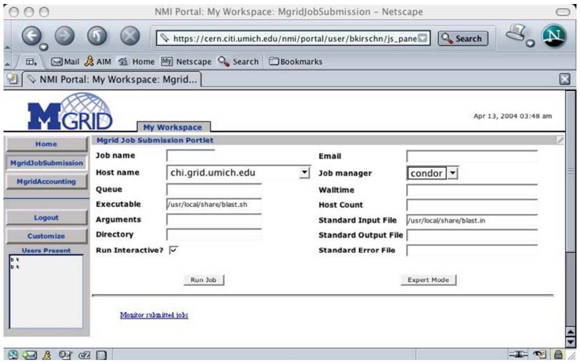
A. User Authentication

X.509 is the defacto standard for grid identity management and authentication. However, X.509 is not the primary method used by sites for managing user identity. Some sites rely on NIS [22], others on Microsoft Active Directory [23], or on a centralized Kerberos principals database. The University of Michigan operates a comprehensive user identity management system based on Kerberos.

Grid users must be members of administrative domains that can verify and vouch for the identity and attributes of a user. Administrative domains that take on this task must be able to maintain X.509 user identities for sponsored users, since X.509 is the defacto standard for grid computing environments. There are two paths that a site can take to manage these identities: create and manage a "shadow" X.509 identity system that parallels, but does not rely on, the existing identity system; or make use of an approach that can "bridge" from existing identities to X.509 identities.

The University of Michigan developed KX.509/KCA [4] to provide a bridge from the production Kerberos [24] identity management system that has been in place for over 10 years. The University of Michigan maintains a *Kerberos Certificate Authority* that provides institutional validation for the X.509 certificates generated by KX.509/KCA.

Reliance on an existing user identity management system provides many benefits. Groups that supply resources for a grid are assured that users who request resources have been properly authenticated, which decreases the risks inherent in participating in a grid. Institutional costs for participating in a grid are reduced by eliminating the need to operate a shadow user identity management system. Other web-based campus services can also make use of the X.509 bridge solution. KX.509/KCA was originally developed at the University of Michigan for supporting secure access to web-based university services. By reducing the complexity of user authentication, fewer passwords and authentication mechanisms are necessary, which simplifies life for the user and improves the overall level of security for the institution.



Mgrid Job Submission Portlet

Job name
Host name chi.grid.umich.edu
Queue
Executable /usr/local/share/blast.sh
Arguments
Directory
Run Interactive? ☑
Email
Job manager condor
Walltime
Host Count
Standard Input File /usr/local/share/blast.in
Standard Output File
Standard Error File

Fig. 2. MGRID OGCE Portal

In *Walden*, the user loads the Kerberos 5 client and the KX.509/KCA software on their laptop or desktop computer. The user then authenticates to the university Kerberos server (using a platform-specific Kerberos client or $kinit$), and launches the KX.509/KCA client. KX.509/KCA securely contacts a shadow Kerberos Key Distribution Center (KDC) that returns an X.509 certificate that is signed by the University of Michigan Kerberos Certificate Authority. The user's password is ***never*** transmitted over the network (in any form) during this authentication process. The client software required for user authentication is supported and maintained by the University of Michigan central information technology support group. Once the user has a valid X.509 certificate, they can use MGrid either with a locally installed globus client, or by using the MGrid grid computing portal. Figure 2 shows the MGRID OGCE portal.

The next section describes how *Walden* maps the X.509 grid identity into a local form of identity on the remote resource.

B. Establishing Local Identity and Authorization

Once a user has been properly authenticated and presents a resource request with a signed X.509 certificate, the next step is to match the user with a identity local to the remote resource, and to ensure that the user is authorized to make use of the resource.

Matching the user with a local identity is complicated by the fact that very few resources can natively operate using X.509 certificates, which can make use of X.509 for authorization and access control lists (ACLs). To address this issue, we needed a mechanism to securely convert the X.509 identity into an acceptable form of local identity.

Walden makes use of the Globus authorization callout feature described in the previous section to perform the conversion from X.509 based identity to local identity and enforce authorization policies.

To implement these mechanisms, we developed an authorization server that uses the open source Sun Microsystems

XACML implementation [12].

XACML defines a powerful policy engine that allows definition of sets of policies, which are further divided into targets, rules, conditions and obligations. Policy sets and rules are evaluated and combined based on predefined algorithms (*Deny-overrides, Permit-overrides, First-applicable, Only-one-applicable*). Targets, rules, and conditions act on attributes which may be part of the original request, or may be retrieved based on a defined "Attribute Authority" (AA)[1] Obligations can provide further constraints on an authorization decision [11].

The user's X.509 identity can be converted to a local account that is permanently assigned to a user, or to a temporary guest account. A pool of local guest (template) accounts are defined for each resource to provide access for authorized users that do not have a permanent local account. With a site defined *grace period* [8] that determines the length of time that the guest account binding ("lease") will be maintained after last use, a grid user can continue to use the same local guest user account between successive job runs. A record of the binding between an X.509 and local account identity must be retained indefinitely for accounting and security purposes.

Figure 3 describes the authorization components and the communication between the components. Each step in the authentication and authorization process is shown in the figure and is described below.

1) To authenticate, the user performs a $kinit$ (or GUI equivalent) to obtain a Kerberos 5 Ticket Granting Ticket (TGT) and invokes kx509 to obtain a signed X.509 certificate based on the Kerberos 5 TGT.
2) After the user has properly authenticated, they may gain access to grid resources. Using a local Globus client, or interacting through a grid web portal, the user requests access to grid resources. This request results in the invocation of the $globusrun$ command, which will send the request to an appropriate Globus gatekeeper daemon process.
3) Upon receiving a request, the Globus gatekeeper normally searches the $grid-mapfile$ for the X.509 DN of the requestor in order to find an appropriate authorized local identity (e.g. UNIX username). In our implementation, the gatekeeper uses the globus_gridmap_callout function to send an authorization request (user X.509 DN, resource, service) to the *Walden Authorization Server*, which serves as a bridge between the C-based globus_gridmap_callout function, and the Java-based XACML policy engine. The *Walden Authorization Server* is a daemon running on the gatekeeper. It receives and forwards the authorization request to the PEP (Policy Enforcement Point). The PEP formats an XACML request, and sends it to the PDP (Policy Decision Point). Though this implementation makes use of an Attribute Authority to retrieve group membership attributes, group membership could additionally be retrieved from an X.509 AC, to support more ad-hoc virtual organizations of users.

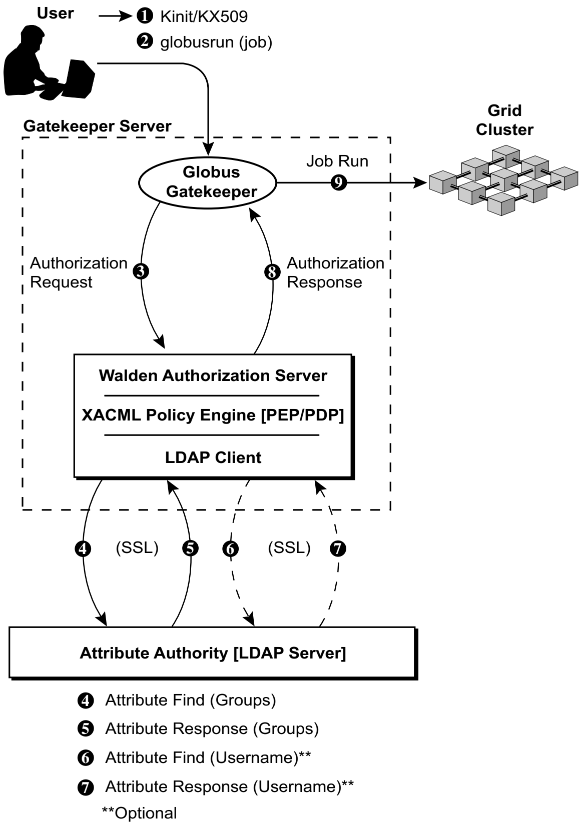

User
❶ Kinit/KX509
❷ globusrun (job)
Gatekeeper Server
Grid Cluster
Globus Gatekeeper
Job Run
❾
Authorization Request
❸
❽
Authorization Response
Walden Authorization Server
XACML Policy Engine [PEP/PDP]
LDAP Client
❹
(SSL)
❺
❻
(SSL)
❼
Attribute Authority [LDAP Server]
❹ Attribute Find (Groups)
❺ Attribute Response (Groups)
❻ Attribute Find (Username)**
❼ Attribute Response (Username)**
**Optional

Fig. 3. *Walden* Authorization Process and Components

4) Next, $Walden$ must determine if the user has authorization to access the requested resource. The PDP (Policy Decision Point) applies the authorization request against a configurable policy or policy-set located on the local hard disk or LDAP server. The SUN XACML policy engine invokes an AttributeFinder, which contacts an Attribute Authority to locate any attributes missing in the original request.
5) The AttributeAuthority (via a secure LDAP connection) returns a "bag of attributes", corresponding to the groups in which a user has membership, to the XACML AttributeFinder component. The Attribute Authority can be defined as an existing campus LDAP server, to use pre-existing user groups, a grid-specific LDAP server with user groups defined by resource owners, or both.
6) The PDP (Policy Decision Point) determines if the user is in an authorized group, as defined in the policy, and applies any other defined rules and conditions in the policy. The PEP (Policy Enforcement Point) receives the decision and any optional obligations from the PDP. The local user identity is derived from an obligation to bind the user to a guest account (described in detail

[1] An *Attribute Authority* is an entity that is trusted by at least two entities to create and assign attribute certificates. An *Attribute Certificate* is a set of attributes and a public key certificate identifier that are made unforgeable by use of the digital signature created with a private key.

in the next section) or an AttributeFind to retrieve the user's permanent local user identity from the Attribute Authority.

7) If a permanent local user identity is defined, the Attribute Authority (LDAP Server) returns the user's permanent local identity to the PEP.
8) If the user is authorized, the permanent or guest local user identity is returned to the *Walden Authorization Server*, which forwards it on to the globus gatekeeper.
9) At this point, the user has been bound to a local identity and as been granted access to the resource by the *Walden Authorization Server*. The gatekeeper can then proceed to contact the resource on behalf of the user and release control to the user or initiate the computational task.

C. Binding Users to Local Userids

When a user requests access to resources on the grid, the grid form of identity must be converted into a identity that is local to the resource (e.g. a UNIX username). Traditionally, the model for this conversion is that an *a priori* arrangement is required that creates a permanent one-to-one mapping between a user and a local resource user identity. Unfortunately, this model does not easily scale: it is difficult to manage the local mapping in a timely, secure, and accurate manner. Moreover, the preallocation of resources (such as disk space for home directories) for these permanent bindings is wasteful if a significant percentage of the user community never makes use of the resource.

$Walden$ addresses this issue by providing a temporary one-to-one binding between a user and a local identity. The temporary binding selects a special guest account from a pool of guest accounts, and assigns the account to the user for the period of use plus a *grace period* that facilities reuse of the account over a period of time. Although a user may be bound to a particular local or guest account, users must still be authenticated and authorized for each and every request.

Users with permanent and temporary guest local user accounts are distinguished and configured through membership in separate user groups referenced in the XACML policy. Those users who are authorized, but lack local permanent accounts, are permitted access using a policy which makes use of XACML *obligations*. An *obligation* consists of an attribute or set of attributes that define an action which the PEP is 'obliged' to perform in order to honor the authorization decision the PDP has made. $Walden$ uses these obligations to define a set of temporary guest accounts that can be temporarily bound to user X.509 identities.

Guest user accounts are defined in the XACML policy by a template basename (e.g. 'guestuser') and an iteration count (e.g. '30'), which are combined to create a predefined guest account (e.g. 'guestuser30'). Guest accounts that are in use are cached in the PEP, which maintains a persistent mapping between the user's X.509 DN, the guest user account name, and the current state of the guest account. A guest account is considered busy and unavailable for use by others if the guest account was recently used to submit any jobs that are either pending or executing on the local cluster. An account is also considered busy and unavailable if the period of time since the last use of the account does not exceed a configurable *grace period*, which is usually several days in length, but may last up to several months.

The number of guest accounts required, along with the duration of the *grace period* can be calculated based on the historic usage patterns of each cluster, along with occasional monitoring of logs [8]. Monitoring allows the site administrator to ensure that $Walden$ is providing an acceptable level of service (accepted or rejected guest account requests) to the grid community.

D. Scalable Account Management

To this point, we have described the process by which a user gains local access to a resource. If a site administrator must maintain n permanent user accounts in a domain of m hosts, the cost for managing user accounts is $O(mn)$. We were concerned about reducing this administration cost, to allow a site to make use of grid computing without significantly increasing administrative costs. To reduce these costs, we made use of the $nsswitch.conf$ (Name Service Switch config) facility on UNIX to manage password files. Essentially, we configured our Linux systems to make use of the nss_ldap [25] module to retrieve the password file entries for guest (and other) users from the secure LDAP server used for the Attribute Authority. We used the posixAccount LDAP object class defined in RFC 2307 [26] to store the account information required to replace the password file entry.

This mechanism allows a site administrator to configure their hosts only once to refer to an LDAP server for account entries that are not maintained in the local password file. The administrator can then manage local grid accounts in the LDAP server one time for all of the systems that make use of nss_ldap. This reduces the ongoing administrative cost to $O(m)$, for m accounts.

E. Guest Account Issues

Resource usage can be tracked and logged for accounting purposes because the distinguished name (X.509 DN) of the user bound to a guest account is returned to the Globus $jobmanager$. The $jobmanager$ preserves the DN by inserting it into the scheduler job control file (job submission script). The scheduler then records the DN in the scheduler job log for post processing by accounting software (for example, PBS XML Accounting [27]).

Guest accounts can also make use of network file space defined for a persistent network user id by making use of nsswitch.conf redirection with LDAP to manage automounter NFS map files.

Using these mechanisms, $Walden$ can accommodate both permanent users and guest grid users without wasting valuable resources.

The next section of this paper describes use case scenarios for $Walden$ in several different operating environments.

IV. Use Case Scenarios

One of the strengths of the Walden Authorization model is its flexibility to support both large institutional organizations, multi-institutional organizations, and small dynamic virtual organizations. The following use cases illustrate several different user communities needing to access grid cluster resources.

A. Institutional Grid Cluster

A computational cluster is made available to the grid computing community within the university. The owner of the cluster wants to restrict usage to university users to initiate limited term jobs during specified time periods. The cluster owner uses the existing university directory service to define a group of users authorized to use the cluster, or alternatively, uses an existing group or set of groups already defined in the university directory service. The cluster owner uses software maintained by MGRID to create an XACML policy file that specifies the authorized groups and the restricted time periods of use. The head node of the cluster or a gatekeeper host is loaded with the Globus Toolkit, Walden Authorization, and a job scheduler client to allow the globus gatekeeper to submit jobs to the computational cluster.

Users of the campus grid create a temporary KX.509 certificate after authenticating using their institutional Kerberos identity. This user's X.509 certificate is verified as part of the trust domain of the gatekeeper. The user must be a member of the group that has been granted access by the resource owner to gain authorization to a requested resource. The user can be dynamically added or removed from this user group by the resource owner or any delegate authorized to edit the LDAP user group. This user does not have or need a local account on the grid cluster. The XACML policy defines a pool of 20 guest user accounts that can be mapped to authorized users.

The user is able to submit a series of Blast searches, evaluate the data upon completion, and submit additional Blast searches using the $same$ local user identity. The defined grace period retains that user's local identity between job submissions.

At the end of the month, the resource administrator is able to query the usage over the past month, broken down by users identified by their X.509 Distinguished Name.

B. Multi-Institutional Grid Cluster:

A cluster of linux servers is needed to serve a large international, multi-institutional physics endeavor. Some users require static local identities, but most will share a pool of users with a generous grace period. MGRID provides an LDAP server allowing the resource owner to define a *groupOfNames* listing each user's X.509 distinguished name. The resource owner and MGRID together create an XACML policy file which references the specified user group (using a defined attribute type that will be used to locate the correct LDAP server as the Attribute Authority).

Users are authenticated at the gatekeeper using X.509 certificates from a diverse assortment of trusted CA (certificate authorities). In order to be authorized, the user must be a member of one of the user groups defined by the resource owner. If the user is a member of a group with local accounts on the grid cluster, the gatekeeper will authorize the user with the specified local user id. If the user is a member of a group without local accounts on the grid cluster, the user will be mapped to a pool of temporary user accounts as described above. Resource administrators will be able to query usage, as above, using the X.509 DN to identify users.

C. Dynamic Virtual Organizations:

Grid cluster users may also be within a fluid, virtual organization. Grid clusters can be made available to these users as well, using X.509 ACs (attribute certificates) signed by a trusted certificate authority. The XACML policy file defines access that references a user group defined by an X.509 Attribute Certificate (using a defined attribute type). This flexibility could be used to authorize users from trusted institutions that generate X.509 AC for their local authorization model.

V. Implementation Status

We have implemented and successfully tested all of the components described in Figure 3 along with the dynamic mapping of users to guest account. We have successfully tested replacing the password file using the nss_ldap nsswitch mechanism. We are working on the initial deployment of $Walden$ on the ATLAS cluster in the Physics department and on a Macintosh OS-X cluster managed by the university Information Technology Central Services unit. We plan to use $Walden$ for the ATLAS Data Challenge II to demonstrate its usefulness for assigning guest accounts to external users.

We are planning to develop an XACML policy file editor as a grid portal portlet to allow resource administrators to manage access policies.

VI. Conclusion and Future Work

The WALDEN authorization model described in this paper meets or exceeds the stated design goals. The use of group based access policies allows for a solution which is scalable to tens of thousands of users or more. Administration of access groups is independent from the authorization package itself, and can even make use of existing Directory Services, such as a university's existing LDAP server for staff, faculty and students. The XACML policy engine provides a powerful and extensible policy definition language, which supports retrieving attributes from specified attribute authorities along with the definition of extension functions and primitive types. Resource owners have complete control over who is a member of groups they define as having access to their resources. Users can be bound to a (temporary) local user id from an available pool of guest local user identities. The binding between user and local user id remains active for as long as the user's jobs are active, including a configurable ”grace period”. Virtual Organizations can be created, modified and destroyed by defining membership within a defined Attribute Authority.

The current implementation of the Walden Authorization model keeps the Policy Enforcement Point (PEP) and Policy Decision Point (PDP) on the same host as the gatekeeper.

While this provides for quick and efficient local communication, it does not provide easy access to the authorization services from other hosts. Future development of the Walden Authorization model may support providing web services allowing access for applications such as meta-schedulers, as well as flexibility in separating instantiating the PEP and PDP on separate hosts.

ACKNOWLEDGMENT

The authors would like to thank the members of the MGRID Executive Committee (Dr. Homer Neal, Dr. William Martin, Dr. Thomas Finholt, Jeff Ogden, Dr. Brian Athey, and Scott Gerstenberger) and the Technical Leadership Team (Dr. Shawn McKee, Roy Hockett, Eric Hofer, Dr. Charles Severance, Dr. Abhijit Bose) and the Center for Information (Olga Kornievskaia, Kevin Coffman) and Information Technology Central Services (Dr. James Hilton, William Doster, Robert Riddle) for their ongoing support, guidance, and contributions to the MGRID effort.

REFERENCES

[1] I. Foster and C. Kesselman, “Globus: A metacomputing infrastructure toolkit,” *International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, 1997.

[2] W. A. Doster, Y.-H. Leong, and S. J. Matteson, “Uniqname overview,” *Proceedings of the Fourth Large Installation System Administrator’s Conference (LISA IV) (USENIX Association: Berkeley, CA, 1990)*, p. 27, 1990.

[3] O. Kornievskaia, P. Honeyman, B. Doster, and K. Coffman, “Kerberized credential translation: A solution to web access control,” in *USENIX SECURITY 01*, 2001, pp. 235–250.

[4] B. Doster. (2004) Kerberos leveraged PKI. [Online]. Available: http://www.citi.umich.edu/projects/kerb“˙pki/

[5] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, “Security for Grid services,” in *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, June 2003. [Online]. Available: http://www.globus.org/Security/GSI3/GT3-Security-HPDC.pdf

[6] C. C. I. T. T. Recommendation X.509, *The Directory-Authentication Framework*, C. C. I. T. T., Dec. 1988.

[7] G. Project. (2004) Globus Authorization and Mapping Callout Package. [Online]. Available: http://www-unix.globus.org/security/callouts/

[8] T. J. Hacker and B. D. Athey, “A methodology for account management in grid computing environments,” *Lecture Notes in Computer Science*, vol. 2242, 2001.

[9] OGCE:. (2004) Open grid computing environment. [Online]. Available: http://www.ogce.org

[10] CHEF:. (2004) Comprehensive collaboraive framework project. [Online]. Available: http://chefproject.org

[11] OASIS XACML Technical Committee. (2004). [Online]. Available: http://www.oasis-open.org/committees/tc“˙home.php?wg“˙abbrev=xacml

[12] S. Microsystems. (2004) Xacml project. [Online]. Available: http://sunxacml.sourceforge.net/

[13] D. W. Chadwick and A. Otenko, “The PERMIS X.509 role based privilege management infrastructure,” *Future Generation Computer Systems*, vol. 19, no. 2, pp. 277–289, Feb. 2003.

[14] M. Lorch, D. Adams, D. Kafura, M. K. A. Rathi, and S. Shah, “The PRIMA system for privilege management, authorization and enforcement in grid environments,” in *Fourth International Workshop on Grid Computing - Grid 2003*, Phoeniz, AZ, Nov. 2003.

[15] Internet2. (2004) Shibboleth project. [Online]. Available: http://shibboleth.internet2.edu

[16] R. Alfieri, R. Cecchini, V. Ciaschini, L. Dell’Agnello, A. Frohner, A. Gianoli, K. Lorentey, and F. Spataro, “VOMS, an authorization system for virtual organizations,” in *First European Access Grids Conference*, Santiago, Chile, Feb. 2003.

[17] J. Novotny, S. Tuecke, and V. Welch, “An online credential repository for the grid: MyProxy,” *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, Aug. 2001.

[18] M. J. Litzkow, M. Livny, and M. W. Mutka, “Condor - A Hunter of Idle Workstations,” in *Proceedings of the 8th International Conference on Distributed Computer Systems*. Washington, D.C.: IEEE Press, June 1988, pp. 104–111.

[19] A. B. R. Buyya, “Gridbank: A grid accounting services architecture (GASA) for distributed systems sharing and integration,” *Distributed, Parallel, and Cluster Computing; C.2.4*, Oct. 01 2002. [Online]. Available: http://arXiv.org/abs/cs/0210002

[20] V. Sehkri, I. Mandrichenko, and D. Skow, “Site authorization service (SAZ),” *Distributed, Parallel, and Cluster Computing; D.4.6; K.6.5*, June 16 2003, comment: Talk from the 2003 Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, CA, USA, March 2003, 3 pages, PSN TUBT007. [Online]. Available: http://arXiv.org/abs/cs/0306100

[21] R. Lepro. (2004) Cardea: Dynamic access control in distributed systems. [Online]. Available: http://www.nas.nasa.gov/Research/Reports/Techreports/2003/nas-03-020-ab%stract.html

[22] R. Ramsey, *All About Administering NIS+*, 2nd ed. Upper Saddle River, NJ 07458, USA: Prentice-Hall, 1994. [Online]. Available: http://www.sun.com/books/catalog/ramsey/index.html

[23] J. Streicher-Bremer. (2004) Linux active directory integration how-to. [Online]. Available: http://jaxen.ratisle.net/˜jj/nss˙ldap-AD˙Integration˙how-to.html

[24] J. G. Steiner, B. C. Neuman, and J. I. Schiller, “Kerberos: An Authentication Service for Open Network Systems,” in *Winter 1988 USENIX Conference*. Dallas, TX: USENIX Association, 1988.

[25] PADL Software Pty Ltd. nss_ldap linux module. [Online]. Available: http://www.padl.com/OSS/nss“˙ldap.html

[26] L. Howard, “An approach for using ldap as a network information service,” RFC 2307, Mar. 1998.

[27] R. Mach, “The pbs accounting toolkit,” *SysAdmin*, vol. 12, no. 10, Oct. 2003.