Distributed Web Security for Science Gateways

Jim Basney, University of Illinois, jbasney@illinois.edu
Rion Dooley, University of Texas, dooley@tacc.utexas.edu
Jeff Gaynor, University of Illinois, gaynor@illinois.edu
Suresh Marru, Indiana University, smarru@indiana.edu
Marlon Pierce, Indiana University, marpierc@indiana.edu

ABSTRACT
Science gateways broaden and simplify access to cyberinfrastructure (CI) by providing advanced interfaces to collaboration, analysis, data management, and other tools for students and researchers. As these science gateway interfaces to cyberinfrastructure grow in popularity, web portal developers adopt ad hoc approaches to the security challenges of authentication, authorization, and delegation. Science gateways integrate cyberinfrastructure resources on the researcher's behalf, i.e., accessing data, compute cycles, instruments, and other valuable resources. Resource access often requires use of the researcher's security credentials, in some cases exposing the researcher's long-lived password to potential compromise at the science gateway. There is no standard approach for a researcher to control and limit a science gateway's access to his or her resources. Thus, researchers are required to accept unnecessary risks when using science gateways.

The "Distributed Web Security for Science Gateways" project is addressing these risks by providing authorization and delegation software for science gateways that complies with the Internet Engineering Task Force's standard OAuth protocol. The project is developing an OAuth server implementation and a set of client libraries and authentication modules to enable out of the box integration with common Web platforms, in coordination with gateways and cyberinfrastructure providers. In this paper, we introduce the project, including our planned software architecture.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *authentication.*

General Terms
Security

Keywords
Science Gateways, OAuth

1. INTRODUCTION
Science is migrating from the command-line to the Web. Web browsers have become the most common interface to cyberinfrastructure. In a recent survey of 5,000 NSF PIs, NSF's Campus Bridging Task Force found that "the most common method of accessing CI across the entire dimension of providers is via Web browser/portal" [1]. Science gateways and Web portals are important mechanisms for broadening and simplifying cyberinfrastructure use. Gateways use Web technologies to provide user interfaces and user-supporting services for information, collaboration, scientific application management, data management, and workflow composition and orchestration for both Grids and Clouds [42]. In the US, the XSEDE Science Gateway program and the closely related XSEDE User Portal are notable examples of Web-based access to cyberinfrastructure, with additional support from the NSF's SDCI and SI^2 programs. The Science Gateway tracks at recent TeraGrid conferences and the Grid/Gateway Computing Environments workshops [7][8] provide an overview of these efforts. International gateway efforts are surveyed in the recent International Workshop on Science Gateways [18].

Researchers are attracted to science gateways because they offer a convenient interface for accessing research data, collaborating with colleagues, running simulations, and performing data analysis, without the need for a strong understanding of the underlying cyberinfrastructure intricacies. While gateways are often composed of multiple distributed components, users expect the gateway to broker all of the underlying interactions and support long-running science workflows without requiring constant monitoring and interaction by the user. Users further expect gateways to integrate with other resources in the cloud that are available via standard APIs and protocols (for example, Representational State Transfer (REST) [30]).

Trust is essential for science gateways. Users demand that their research and personal information are kept protected, and resource providers demand that their computing and storage resources are used appropriately by vetted users. Meeting these requirements has always required a leap of faith by both gateway users and resource providers [15][16], as gateways are not centrally managed or reviewed by any authority, and the distributed nature of the more sophisticated gateways introduces credential delegation and renewal challenges that gateways today must solve individually. Furthermore, Web architectures have shifted dramatically over the last five years, pulling cyberinfrastructure in new design directions [6]. Science gateway security must evolve to embrace important trends such as Web gadgets and REST-based Web services.

In this paper we introduce our new "Distributed Web Security for Science Gateways" project, which will enable both software developers and cyberinfrastructure providers to deliver usable and trustworthy science gateways using the widely adopted Internet Engineering Task Force (IETF) standard OAuth protocol [25][32] to support distributed authentication, delegation, and authorization. To accomplish this, we are developing a standards-compliant OAuth service implementation to securely delegate, deliver, and renew credentials to science gateways on a user's behalf. This solution will be flexible enough to integrate with the MyProxy online credential repository [23] as well as other credential management and authentication services. We are also developing client libraries for Java, Python, and PHP as well as authentication modules for popular Web frameworks used by today's research communities.

This paper is organized as follows. In Section 2 we present the motivating concept for our work. In Section 3 we describe related work. In Section 4 we detail our project components and release timeline. In Section 5 we discuss expected impacts for science gateways. We conclude in Section 6.

2. MOTIVATING CONCEPT

We now illustrate the core concept of our project, developed in greater detail in Section 4. As discussed above, science gateways broker access to resources on behalf of users, and this requires security credentials. A common approach is for the gateway to obtain a certificate on the user's behalf from a MyProxy server. Figure 1 below illustrates this interaction.

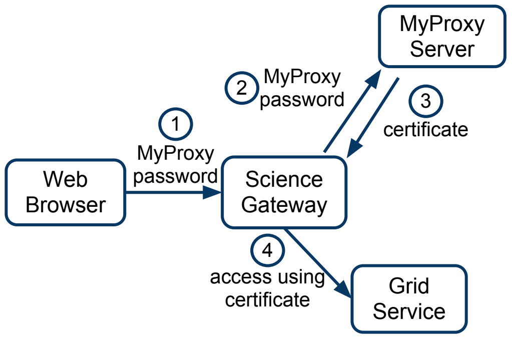

Figure 1. MyProxy use by gateways today.

At the gateway, the user provides a MyProxy username and password (step #1 in Figure 1). The gateway then submits these to MyProxy (step #2) and, based on that authentication, obtains a short-lived certificate (valid for a few hours or days) (step #3). Once the certificate is loaded into the gateway, the user can submit jobs, transfer files, and perform other operations through the gateway using his or her accounts on grid services (Globus GRAM or GridFTP servers, for example) [11][12][13]. The science gateway block in Figure 1 is commonly a collection of services on multiple machines (for example, the UltraScan gateway, discussed below); in this case the portal may delegate the certificate to a supporting service.

This approach has several risks: 1) Submitting a long-lived password to the gateway exposes the password to potential compromise by a malicious or subverted gateway. 2) The certificate delegation procedures required by multi-component gateways are not standardized and are implemented by each gateway as it deems best. 3) Gateway software, deployments, and protocols do not generally undergo security assessments. Conducting individual gateway security reviews is not scalable; instead, gateways should be provided with high quality libraries and services that have passed security assessments.

We are addressing these risks by implementing an OAuth interface to MyProxy. Figure 2 illustrates the foundation of our approach. By providing a trusted MyProxy OAuth service, users can allow gateways to obtain short-lived credentials to act on their behalf without having to trust the gateways with their infrastructure-wide passwords. On this foundation, we can provide additional security services needed by gateways, including distributed delegation to multiple services, credential renewal, simplified integration with external authentication (such as InCommon), and security tools for the growing array of REST-based CI services.

We now review the steps for OAuth-enabling MyProxy. Additional features are described in Section 4. When the user accesses the gateway and wants to perform actions that require credentials, the gateway redirects the user's browser (step #1 in Figure 2) to the trusted MyProxy Web interface (OAuth front-end) where the user enters his or her MyProxy password and approves the transaction. In addition to providing a higher level of trust both for the user and the resource providers, the MyProxy Web interface can, at the user's request, set a browser cookie to provide single sign-on for this OAuth transaction across different portals. Then (step #2), the OAuth front-end redirects the user back to the portal with a time-limited OAuth token ("authorization code" [25]) that allows the science gateway to get a certificate for the user. The gateway connects to the OAuth service and exchanges the token (step #3) for a certificate (step #4). The gateway can then access grid services on the user's behalf using the certificate (step #5). Secure delegation, renewal, and general REST security can be built on this foundation, as described in Section 4.

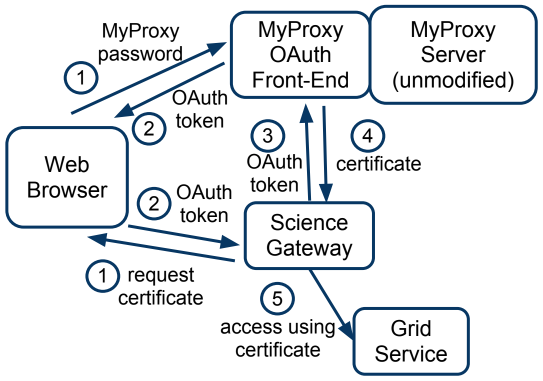

Figure 2. Using OAuth with MyProxy.

3. RELATED WORK

Our work is building on a significant body of existing software, protocols, and standards. These building blocks are described below.

OAuth: OAuth provides a relatively well-understood and powerful mechanism for resource owners (that is, end users) to allow controlled, third-party access to their resources. This requirement is common across many applications, including the science gateways and web portals that are the focus of our work. As illustrated in Figure 2, the resource owner interacts with an OAuth authorization server to approve access to the resource by the third party (the OAuth client) and issue tokens to the third party for secure access to the resource. The use of separate OAuth tokens enables the resource owner to allow access without exposing his or her resource credentials (for example, passwords) to the third party. OAuth tokens can be for one-time use or renewable, can have limited lifetimes, and can limit the scope of access to the resource in various ways. The OAuth 1.0 protocol was initially developed in 2007 and was published as an IETF informational RFC [32] in April 2010. An update to the protocol, OAuth 2.0 [25], is under development in the IETF OAuth working group, and we are migrating to the OAuth 2.0 standard. Open Source OAuth implementations are widely available, and OAuth support is included in many Web frameworks. We are building on existing OAuth libraries.

OAuth is a very flexible security protocol, and its strength depends on how it is used. The management of OAuth keys and secrets and the methods employed for authentication and signing critically impact the overall security of the system. For example, the widely reported compromise of Twitter's use of OAuth in

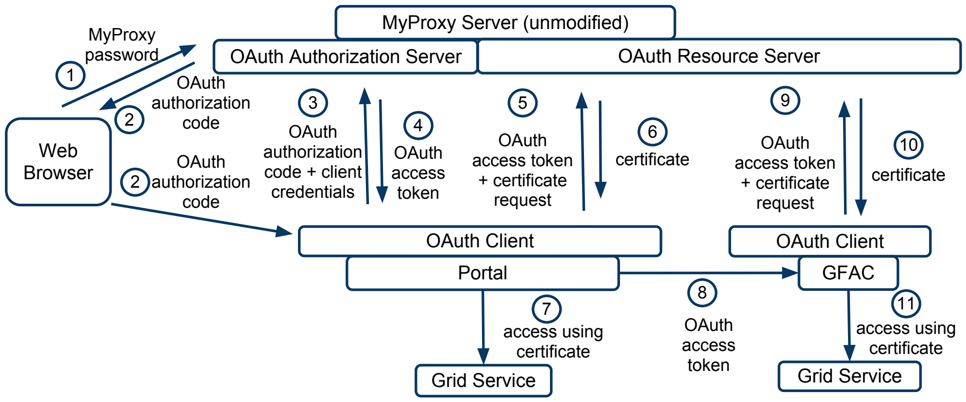
MyProxy Server (unmodified)
OAuth Authorization Server
OAuth Resource Server
1 MyProxy password
2 OAuth authorization code
Web Browser
2 OAuth authorization code
3 OAuth authorization code + client credentials
4 OAuth access token
5 OAuth access token + certificate request
6 certificate
9 OAuth access token + certificate request
10 certificate
OAuth Client
Portal
OAuth Client
GFAC
7 access using certificate
8 OAuth access token
11 access using certificate
Grid Service
Grid Service

Figure 3. Credential delegation via OAuth resource server.

September 2010 was a result of weaknesses in how Twitter employed the OAuth protocol, rather than a vulnerability in OAuth itself [43]. As another example, the question of message signatures versus bearer tokens has been an active topic of discussion in the OAuth 2.0 working group, in part regarding trade-offs between implementation complexity and threat protection in different scenarios [5][26]. The security profile of an OAuth use case depends on many factors including the type of resource being protected, contacting a well-known service endpoint versus doing dynamic service discovery, and whether the user agent is a Web browser, native application, or autonomous service. Based on our experience with CILogon and TeraGrid/XSEDE, we believe that OAuth is sufficiently flexible and extensible to support science gateway security requirements. As we fit the OAuth protocol to different science gateway use cases, we will adhere to the guidance of OAuth security considerations documents, as well as public peer review (for example, the XSEDE security working group) and independent vulnerability assessment by the University of Wisconsin MIST (Middleware Security and Testing) team [19][20].

MyProxy: MyProxy [22][23] combines a credential repository with an online certificate authority to allow users to securely obtain public key infrastructure (PKI) credentials when and where needed for access to secure cyberinfrastructure services. MyProxy is mature software that is widely used across science gateways and portals in XSEDE and elsewhere to obtain certificates for users based on username/password authentication. MyProxy's password-based protocol was specified in 2005 in a Global Grid Forum Experimental Document [10]. The main drawback to this protocol is that it requires the user to provide a password to the gateway for use with MyProxy, which our work will address.

XSEDE OAuth Service: The XSEDE OAuth Service (initially developed for TeraGrid) provides OAuth 1.0 access to the XSEDE MyProxy server for science gateways. We are now developing a general-purpose OAuth 2.0 service for MyProxy, which can be deployed by XSEDE as well as other MyProxy sites. Our TeraGrid conference paper [39] describes the OAuth 1.0 implementation in detail, including a discussion of security considerations. When we update to OAuth 2.0, we will publish updated implementation details, including updated security considerations.

CILogon: The CILogon Service [3] allows users to authenticate with their home organization through the InCommon Federation to obtain a certificate for secure access to cyberinfrastructure. The service is implemented using technologies from the Shibboleth, MyProxy, and GridShib projects. One of the interfaces supported by the CILogon Service enables certificate delegation to portals via the OAuth 1.0 protocol. The portal redirects the user to the CILogon Service, where he or she authenticates (via InCommon and Shibboleth at his or her home institution's identity provider) and then approves issuance of a certificate to the portal. Then the CILogon Service redirects the user back to the portal, where the certificate is now available, so the user can submit computations, access data, and perform other tasks that require authenticated access to underlying cyberinfrastructure resources using that certificate. Currently, the implementation of the OAuth capability is closely tied to the operation of the CILogon Service. We are generalizing this capability for use with other services.

OGCE: The Open Gateway Computing Environments (OGCE) [28] project provides software to support science gateways. These components include the OGCE Gadget Container for rendering OpenSocial gadgets; GFAC, a tool for wrapping command line operations as secure services; and the OGCE Workflow Suite, a set of tools for composing, executing, and monitoring scientific workflows. Supporting software includes XRegistry and the OGCE Messaging system. Both UltraScan and GridChem use OGCE for science application and workflow management. Our work will be integrated into the operations of these gateways as well as included in OGCE downloadable software.

In addition to the above building blocks, our work is related to and enables integration with other open source security approaches and solutions.

Shibboleth, SAML, and InCommon: The InCommon Federation enables Web single sign-on (SSO) for the nation's education and research institutions via the OASIS standard SAML Web Browser SSO Profile implemented by the Internet2 Shibboleth software and others. InCommon enables faculty, staff, and students to use their university security credentials to authenticate to online services both internal and external to campus. We do not intend to duplicate or compete with this valuable national capability. Instead, as we have shown with CILogon, OAuth naturally integrates with and complements existing Web SSO solutions. The OAuth authorization server can support a variety of methods for resource owner authentication. Support for campus authentication via InCommon is a high priority capability for our OAuth authorization server implementation, since this will overcome an important hurdle many gateways face when integrating with educational usage. Already in the CILogon project we have integrated OAuth 1.0 with InCommon authentication as discussed above. A significant benefit of this project is that enabling OAuth support in a science gateway enables access via the different (often more complex)

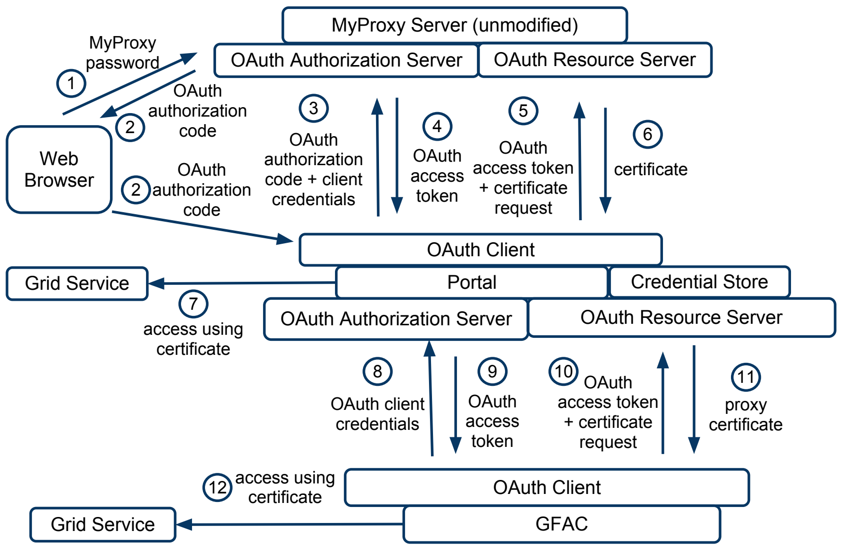

Figure 4. Credential delegation via "2-legged" OAuth.

authentication methods supported by the OAuth authorization server. The OAuth approach avoids integrating these different methods directly into the gateway applications themselves.

OpenID: OpenID [29] provides another standard method for decentralized authentication on the Web. OpenID has been adopted by many commercial providers including Google, VeriSign, and Yahoo!. Since many users already have accounts with these providers, OpenID provides another attractive integration point for science gateway security, particularly in cases where a higher level of assurance authentication is not required. Thus, OpenID is another authentication method to be supported by our OAuth authorization server implementation. An OpenID+OAuth hybrid protocol has been proposed for OAuth 1.0 that streamlines the user experience when the two protocols are used together, and we are tracking evolutions of this work for OAuth 2.0 and OpenID Connect. In the CILogon project we have already experimented with supporting OpenID authentication alongside InCommon campus authentication as an option for users, including integration with OAuth 1.0.

4. PROJECT COMPONENTS

OAuth Interface to MyProxy: As described previously, our first deliverable is the implementation of an OAuth interface to MyProxy. This deliverable builds on existing work in the CILogon project to support portal authentication to the CILogon Service, based on OAuth 1.0, in prototype use by the Ocean Observatories Initiative project. It also builds on prior work for the XSEDE OAuth Service. This deliverable requires updating the existing software to OAuth 2.0 and modifying it to work as a standalone package so that other MyProxy sites can deploy it outside the CILogon Service and XSEDE. The MyProxy OAuth front-end acts as an OAuth Authorization server and can issue tokens that can be used for acquiring certificates. These tokens can be delegated by end users to third party services as well as portal front-ends. We next discuss how we can support certificate delegation on this foundation.

Certificate Delegation via OAuth: Science gateways are composed of multiple, distributed Web entities: Web gadgets from multiple locations are aggregated into common dashboards; gateways may include a mixture of desktop and browser user interfaces; long-running job management and file transfers are implemented as separate network services that are independent of user session cookies; and sophisticated data and metadata management services [17][46] are implemented by services outside the user interface server. End users need an elegant and secure way to delegate credentials among several Web entities that will act on the users' behalf, typically without the user being directly involved.

In our approach, following successful authentication, the end user obtains an OAuth token that can be distributed to all the entities that comprise the science gateway. This includes the user management server (Drupal or the OGCE gadget container, for example), which we call the "portal" for simplicity. Additionally, the end user may authorize the portal to delegate the access token to other entities. As discussed in Section 5.1, UltraScan provides an example: the portal sends the access token to the GFAC service, which uses the token to obtain a certificate from the MyProxy OAuth service. We are exploring two approaches to certificate delegation between different components of the science gateway using OAuth, as illustrated in Figures 3 and 4.

In Figure 3, we show a science gateway consisting of a front-end Web portal and a back-end GFAC application management service (for example, UltraScan). The front-end Web portal obtains an OAuth access token and certificate, so it can access grid services directly on the user's behalf. The portal also shares its OAuth access token with the GFAC service (securely, i.e., over a mutually authenticated TLS connection), so it can also obtain a certificate for the user. In this scenario, the original OAuth service is involved in each delegation among the gateway components, so it can monitor and control the delegation according to its (and the user's) policy. This solution applies when the portal server knows all services that will be involved in a user session and the three-way trust relationships are pre-configured. This also has the advantage of placing minimal required changes on the portal server.

Figure 4 illustrates an alternate approach. In this case, the portal may itself act as an OAuth server to delegate credentials to other gateway components. The portal obtains a certificate for accessing grid services on the user's behalf as in Figure 3 (steps 1-7). Next, the portal allows the GFAC service (acting as an OAuth client) to authenticate with its OAuth client credentials (step #8), previously registered with the portal, to obtain a new OAuth access token (step #9). GFAC can use its access token to obtain (in steps 10-11) a proxy certificate [31] from the portal that is derived from the user's certificate held by the portal. The GFAC service can then use that proxy certificate to access grid services on the user's behalf. This case is useful when we cannot establish *a priori* trust between GFAC and the OAuth Resource Server, but trust can be established between GFAC and the portal server, because they are both operated as part of a single science gateway. This can occur when not all of the components are under the same administrative control or when the additional services are ephemeral.

Secure Access to Gateway REST Services: We are developing our OAuth service so it will be usable with any OAuth-enabled REST service. REST services are of increasing importance to Web development, including gateways [24][27][35][40]. We therefore need to provide a compatible way to address authentication and authorization issues for REST-based cyberinfrastructure. For example, XSEDE provides many useful information services that currently require password authentication to the XSEDE Central Database (i.e., to obtain allocation information, specific machine status information, job status, etc.). While acceptable for interactive applications, this requires passwords to be stored by value-adding services and workflows that need to consume this information without directly interacting with a user.

We are developing OAuth-compliant authorization solutions for REST-based cyberinfrastructure to address the risks and complexities of current *ad hoc* approaches. If cyberinfrastructure developers were to individually implement their own OAuth-compliant solutions without coordination, the result would be a complex mix of approaches for science gateways to integrate, due to the flexible nature of the OAuth protocol framework. By documenting OAuth profiles and providing implementation libraries for common REST-based cyberinfrastructure requirements, we will promote adoption of common approaches, thereby avoiding duplication of effort across different cyberinfrastructure projects and simplifying the integration task across cyberinfrastructure providers.

Our first target use case is the iPlant Foundational API, which currently implements a custom "auth service" that issues security tokens for access to iPlant REST services, including services accessing other services on a user's behalf. By applying an OAuth-compliant solution to the iPlant use case, we plan to enable fine-grained authorization and delegation on a standard platform that can be applied across iPlant APIs and other cyberinfrastructure.

Integration with External Authentication: Our work relies on integration with existing authentication services (identity providers). In Figure 2 we illustrated how the OAuth authorization service could leverage an existing MyProxy server for authentication. (Note the MyProxy server itself may be integrated with a back-end authentication service such as LDAP or Kerberos.) In our other figures so far we have shown that the resource owner authenticates to the authorization service, without providing details.

Figure 5 illustrates how the authentication to the OAuth authorization server may be performed using an external identity provider (for example, via InCommon/SAML or OpenID). The resource owner, using a Web browser, first authenticates at the identity provider (step #1), which issues an authentication assertion (step #2) to the OAuth authorization server, confirming the resource owner's identity. Then, when the resource owner approves the requested authorization, the authorization server issues an authorization code (step #3) and the OAuth protocol proceeds as discussed above.

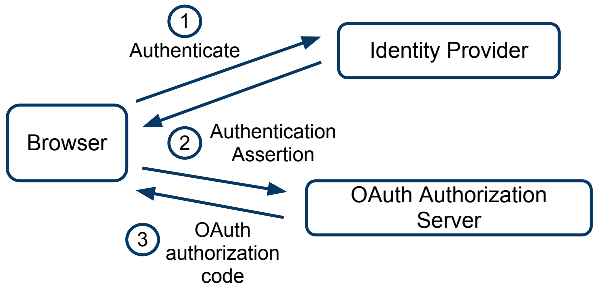

Figure 5. SAML/OpenID authentication.

Figure 6 illustrates an alternative integration scenario, applicable to authentication services (such as LDAP or Kerberos) that are not browser-based. In this case, the resource owner, at a Web browser, provides his or her credentials to the OAuth authorization server (step #1), and the authorization server calls out to the authentication service (step #2) to validate those credentials. Then, the process proceeds as before, i.e., the resource owner approves the requested authorization and the authorization server issues an authorization code (step #3).

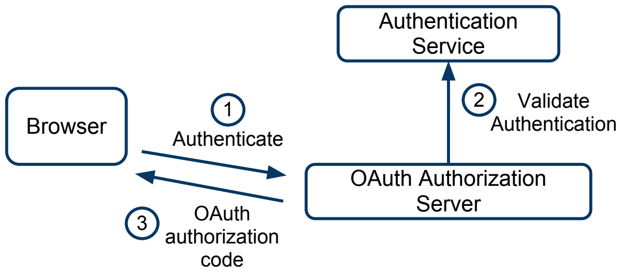

Figure 6. LDAP/Kerberos authentication.

OAuth Credential Refresh: The current OAuth 2.0 draft [25] includes a specification for refresh tokens, which allow an OAuth client to obtain a new access token from an OAuth authorization server when the current access token expires. We believe this functionality maps well to science gateway requirements for securely renewing credentials during long-running workflows [33] and can be implemented in the protocols illustrated in Figures 3 and 4 by issuing refresh tokens along with the access tokens. An OAuth refresh token can allow the gateway to obtain a new certificate from MyProxy or obtain continued authorization to access a REST service, in a controlled and monitored fashion, using a succession of short-lived certificates and access tokens. We plan full support for OAuth 2.0 refresh tokens in our client and server implementations to support these different use cases.

Administrative Functions of the OAuth Service: We plan to support the following administrative functions in the OAuth authorization service: 1) The service will allow gateway administrators to associate external user identities (InCommon principals, for example) with the gateway, govern the acceptance of external authentication mechanisms, and specify a credential (i.e., XSEDE community credential) to associate with an account when users do not have their own individual accounts. 2) The service will enable gateway administrators and resource providers to decorate credentials with additional attributes (for example, as required by XSEDE [38]) through simple interfaces. 3) The service will provide logging and analytics about its use. Finally, 4) the service will support administrative suspension of accounts in case of security incidents.

An Integrated Solution: Taken together, these features will provide a comprehensive solution for distributed authentication, delegation, credential refresh, and authorization for science gateways, compliant with the IETF OAuth protocol. We will provide a server implementation, client libraries, and authentication modules for popular Web frameworks for deployment by cyberinfrastructure providers for use with MyProxy, external authentication services, and REST services.

Timeline: Our project began in August 2011. We expect to make an initial release of the MyProxy OAuth interface by February 2012. We plan initial releases of all software components by August 2012. Then, after operational and performance testing, along with the implementation of additional capabilities based on community input, we plan full, feature complete software releases of all components by August 2013. Our project includes funding through July 2014 for ongoing software testing and support.

5. USE CASES AND PLANNED ADOPTION

The overriding goal of this project is to support scientific research by simplifying secure access to cyberinfrastructure, improving the quality of infrastructure services and software available to gateway developers, and increasing trust of gateways by both users and resource providers. To that end, we are partnering directly with three scientific research projects to establish our initial use cases. The projects are UltraScan, iPlant, and GridChem. We describe each in detail below.

5.1 Engagement with Scientific Research

UltraScan: The UltraScan Science Gateway provides high-resolution analysis and modeling of hydrodynamic data from analytical ultracentrifuges. The gateway is used for fitting experimental data to finite element solutions of flow equations in solution-based studies of biological macromolecules and synthetic polymers. Prof. Borries Demeler, UltraScan's principal investigator, estimates that over 700 biochemists, biophysicists, biologists, and material scientists worldwide rely on UltraScan software for data analysis. By Demeler's estimates, UltraScan has contributed to over 250 peer reviewed publications, including over 30 since 2009 [45]. The software has assisted in the understanding of an array of disease processes, including cancer, neurodegenerative diseases, HIV/AIDS, diabetes, Huntington's disease, and aging studies. UltraScan's popularity and importance in the biophysics community has grown steadily, as has UltraScan's reliance on cyberinfrastructure such as the TeraGrid. In 2008, 151 unique UIltraScan users consumed over 600,000 TeraGrid service units (weighted allocation hours). In 2009, UltraScan consumed over 3 million TeraGrid units. In 2010, the initial allocation of 6 million units was exhausted within 6 months. In 2011, usage patterns shifted as they moved from a pure community account to a model that allowed users to run jobs under their own accounts. As a result, overall usage increased, but usage billable to the community allocation continued at an aggressive pace, burning nearly 500k units through the third quarter.

The UltraScan gateway consists of several components [44]. The components of interest to our work are the Web interface and the job management service. UltraScan develops its Web front-ends with PHP, but it relies on the OGCE GFAC service to manage TeraGrid job submissions [28]. There is thus the need for the gateway to delegate credentials to GFAC after authentication at the portal; the user does not interact with GFAC directly. We discussed this "Certificate Delegation via OAuth" use case in Section 4. UltraScan currently uses a TeraGrid community credential (which provides a workaround), but with UltraScan's increasing popularity there is the need for heavy users to acquire separate allocations.

iPlant: The iPlant Collaborative is a 5-year NSF funded project tasked with building the cyberinfrastructure necessary to support distributed teams of researchers as they attempt to solve key grand challenge problems in the plant sciences. In order to accomplish this charge within the project lifetime, the iPlant CI must leverage as much existing technology as possible and integrate it into a mature suite of APIs and gateways that allows scientists to focus on science rather than technology. iPlant currently has hundreds of users leveraging multiple hardware resources through Web, desktop, cloud, and Web service interfaces (for example: [21]). iPlant is not a single gateway but instead includes a collection of many gateways using a range of approaches. Each interface has its own unique authentication and authorization solution. The Web-based tools use Shibboleth, MyProxy, and LDAP to authenticate users, manage certificates, and manage user identities. The desktop tools use a separate proprietary authentication system tied to the underlying data storage resources. The cloud tools use a mixture of LDAP authentication and their own short-term credential system for managing cloud resources. The Web services currently use SSL, session tokens, HTTP Basic, and HTTP Digest for authentication. Currently, there is no unifying mechanism to handle authentication and authorization across the entire cyberinfrastructure. A goal of our work is to bridge several major gaps in the iPlant architecture that currently necessitate the use of so many different technologies.

Initially, the diverse set of existing iPlant Web-based tools will benefit from our work by gaining a familiar way to authenticate users with a standard set of libraries. Such libraries open the door to easier access for educators, students, researchers, and industry partners through a common authentication framework built primarily on trust and community accepted standards. This means that new tools can be developed and integrated in a shorter period of time, enabling more people to perform more science. Given a common, reusable authentication solution, many of iPlant's current technological challenges disappear. Over time the desktop, cloud, and Web service tools will also benefit from our work by leveraging advanced use cases to enable fine-grained authorization to the three primary iPlant APIs.

GridChem: The Computational Chemistry Science Gateway (GridChem) includes desktop software and middleware services written in Java that integrate molecular science applications and tools for community use. GridChem provides a comprehensive end-to-end solution for managing jobs, workflows, allocations, and data, as well as processing simulation results through a graphical user interface. To date, GridChem has enabled its users to produce results included in 94 publications and 6 dissertations [9].

GridChem users include over 400 researchers, graduate students, and novice users. Users can log in using either their TeraGrid credentials or GridChem-specific credentials. In the former case, the GridChem middleware performs all actions as the user with a certificate retrieved from MyProxy at login. In the latter case, the GridChem middleware uses a TeraGrid community account and tracks all user actions internally using a MySQL database. We discuss use of individual and community accounts by TeraGrid gateways in Section 5.2. GridChem and the related ParamChem project will benefit from our work by moving away from custom to standard solutions. The credential renewal solution is particularly important to GridChem. ParamChem, the next generation of GridChem, with its focus on workflows, requires delegation solutions to enable multiple services to cooperate.

5.2 Engagement with General CI Providers

In addition to partnerships with the scientific research projects and science gateways themselves, the success of our work also depends on integration with general-purpose cyberinfrastructure providers such as XSEDE. Enabling secure coupling between science domain cyberinfrastructure (such as gateways) and general-purpose cyberinfrastructure is a critical component of our work.

TeraGrid/XSEDE: The project team members have been engaged in the TeraGrid Science Gateways program [42] throughout its lifetime and are continuing involvement in the XSEDE gateways program. The XSEDE User Portal receives roughly 11,000 hits a day by over 300 users to check allocations and machine status updates, log in to XSEDE machines, and transfer files. The user portal's underlying information services, some of which require password authentication, are also accessible to gateways [40]. The OAuth-based security framework and libraries that we are developing will provide a consistent mechanism for sharing resources from the user portal to science gateways (and vice versa) without sharing passwords.

Already we have developed an initial OAuth solution for TeraGrid/XSEDE [39] that has now entered production use by Globus Online. XSEDE gateway users enter their password only at the user portal. Then from the portal they can delegate access to their allocated resources for use via gateways. This capability can be further expanded to support other forms of authentication, including InCommon [36]. We believe this capability provides a more secure foundation for linking gateway services with XSEDE.

To support gateway users who do not have their own XSEDE accounts, science gateways use "community credentials" to access XSEDE resources [4][14][37][38]. Supporting both individual credentials (for users with XSEDE accounts) and community credentials (for users without XSEDE accounts) in the same science gateway is complex [2], and few XSEDE gateways currently do it. We plan to include logic in our OAuth service to provide either individual or community credentials to gateways as appropriate, so that gateway developers need only integrate with the single OAuth service to enable both modes of XSEDE resource access.

Globus Online: Globus Online is a high-performance, reliable, secure data movement service (using GridFTP) that integrates with XSEDE and other cyberinfrastructure to enable data movement both within and across cyberinfrastructure deployments. While Globus Online is itself a general-purpose cyberinfrastructure provider, rather than a gateway focused on a specific science discipline, it has the same challenges integrating with XSEDE. Globus Online is an early adopter of the XSEDE OAuth service [39]. Soon the general-purpose MyProxy OAuth service we are developing will enable Globus Online to integrate with additional MyProxy sites via OAuth rather than accepting passwords directly.

6. CONCLUSION

In conclusion, our project is developing open source, standards-compliant authorization and delegation software for science gateways, which addresses the security risks, usability challenges, and implementation complexity of today's *ad hoc* approaches to science gateway security. Through direct engagement with science gateways and other cyberinfrastructure providers, we feel confident that our solutions address current and future science gateway needs and will thereby enhance cyberinfrastructure for research and education.

Please visit www.sciencegatewaysecurity.org for the latest information about our project.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant number 1127210.

8. REFERENCES

[1] J. McGee, V. Welch, G. Almes (eds). National Science Foundation Advisory Committee for Cyberinfrastructure Campus Bridging Task Force Report: Workshop on Software and Services. http://pti.iu.edu/campusbridging

[2] M. Miller, W. Pfeiffer and T. Schwartz, "Creating the CIPRES Science Gateway for Inference of Large Phylogenetic Trees", Gateway Computing Environments Workshop (GCE10), New Orleans, LA, November 2010.

[3] CILogon Service. http://www.cilogon.org/service

[4] K. Price, "Restricted Community Accounts", TeraGrid Conference, June 2006.

[5] E. Hammer-Lahav, "OAuth 2.0 (without Signatures) is Bad for the Web" http://hueniverse.com/2010/09/oauth-2-0-without-signatures-is-bad-for-the-web/ (September 2010).

[6] G. Fox, M. Pierce, "Grids challenged by a Web 2.0 and multicore sandwich", Concurrency and Computation: Practice and Experience 21(3): 265-280 (2009).

[7] Proceedings of the 5th Grid Computing Environments Workshop, http://portal.acm.org/citation.cfm?id=1658260

[8] Gateway Computing Environments 2010 (GCE10) Workshop, http://www.collab-ogce.org/gce10/

[9] Papers and Publications by GridChem Users, https://www.gridchem.org/papers/.

[10] J. Basney, "MyProxy Protocol", Global Grid Forum Experimental Document GFD-E.54, November 26, 2005.

[11] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems", IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[12] R. Butler et al., "A National-Scale Authentication Infrastructure", IEEE Computer 33(12): 60-66 (2000).

[13] V. Welch et al., "Security for Grid Services", Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press, June 2003.

[14] T. Scavo and V. Welch, "A Grid Authorization Model for Science Gateways", International Workshop on Grid Computing Environments, 2007.

[15] TeraGrid Gateway Security Summit, San Diego, January 28-30, 2008.

[16] V. Hazlewood and M. Woitaszek, "Securing Science Gateways", TeraGrid Conference, July 2011.

[17] IRODS: Data Grids, Digital Libraries, Persistent Archives, and Real-Time Data Systems, https://www.irods.org.

[18] International Workshop on Science Gateways, http://agenda.ct.infn.it/conferenceDisplay.py?confId=347

[19] J. Kupsch and B. Miller, “Manual vs. Automated Vulnerability Assessment: A Case Study”, First International Workshop on Managing Insider Security Threats, West Lafayette, IN, June 2009.

[20] J. Kupsch, B. Miller, E. César, and E. Heymann, “First Principles Vulnerability Assessment”, 2010 ACM Cloud Computing Security Workshop (CCSW), Chicago, IL, October 2010.

[21] M. Hanlon et al., “My-Plant.org: A Phylogenetically Structured Social Network”, Gateway Computing Environments Workshop (GCE10), New Orleans, LA, November 14, 2010.

[22] J. Novotny, S. Tuecke, and V. Welch, “An Online Credential Repository for the Grid: MyProxy”, Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001, pages 104-111.

[23] J. Basney, M. Humphrey, and V. Welch, "The MyProxy Online Credential Repository", Software: Practice and Experience, Volume 35, Issue 9, July 2005, pages 801-816.

[24] S. Cholia, D. Skinner and J. Boverhof, “NEWT: A RESTful Service to Build Web Applications for High Performance Computing”, Gateway Computing Environments Workshop (GCE10), New Orleans, LA, November 14, 2010.

[25] E. Hammer-Lahav (ed), D. Recordon and D. Hardt, “The OAuth 2.0 Authorization Protocol”, September 22, 2011. https://tools.ietf.org/html/draft-ietf-oauth-v2

[26] H. Tschofenig and B. Cook, “Thoughts about Digital Signatures for the Open Web Authentication (OAuth) Protocol”, October 18, 2010, https://tools.ietf.org/html/draft-tschofenig-oauth-signature-thoughts

[27] C. Ruby, M. Green and S. Miller, “Orbiter Commander: A Flexible Application Framework for Service-Based Scientific Computing Environments”, Gateway Computing Environments Workshop (GCE10), New Orleans, LA, November 14, 2010.

[28] M. Pierce, S. Marru, R. Singh, A. Kulshrestha, and K. Muthuraman, “Open Grid Computing Environments: Advanced Gateway Support Activities”, TeraGrid Conference, Pittsburgh, PA, August 2-5, 2010.

[29] OpenID Specifications, http://openid.net/developers/specs/

[30] R. Fielding, R. Taylor, “Principled design of the modern Web architecture”, ACM Transactions on Internet Technology, 2(2), p. 115-150, 2002.

[31] S. Tuecke et al., “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”, IETF RFC 3820 (Standards Track), June 2004.

[32] E. Hammer-Lahav (ed.), “The OAuth 1.0 Protocol”, IETF RFC 5849 (Informational), April 2010.

[33] D. Kouril and J. Basney, "A Credential Renewal Service for Long-Running Jobs", 6th IEEE/ACM International Workshop on Grid Computing (Grid 2005), Seattle, WA, November 13-14, 2005.

[34] XSEDE Science Gateways, https://www.xsede.org/gateways

[35] W. Wu, T. Uram, M. Wilde, M. Hereld and M. Papka, “Accelerating science gateway development with Web 2.0 and Swift”, TeraGrid Conference (TG '10), ACM, New York, NY, USA, Article 23, 7 pages.

[36] J. Basney, T. Fleury and V. Welch, “Federated Login to TeraGrid”, 9th Symposium on Identity and Trust on the Internet (IDtrust 2010), Gaithersburg, MD, April 2010.

[37] V. Welch, J. Barlow, J. Basney, D. Marcusiu, N. Wilkins-Diehr, "A AAAA model to support science gateways with community accounts", Concurrency and Computation: Practice and Experience, Volume 19, Issue 6, March 2007.

[38] J. Basney, V. Welch and N. Wilkins-Diehr, “TeraGrid Science Gateway AAAA Model: Implementation and Lessons Learned", TeraGrid Conference, August 2-5, 2010, Pittsburgh, PA.

[39] J. Basney and J. Gaynor, "An OAuth Service for Issuing Certificates to Science Gateways for TeraGrid Users", TeraGrid Conference, July 18-21, 2011, Salt Lake City, UT.

[40] L. Liming et al., “TeraGrid's integrated information service”, 5th Grid Computing Environments (GCE) Workshop, 2009.

[41] J. Basney, S. Martin, JP Navarro, M. Pierce, T. Scavo, L. Strand, T. Uram, N. Wilkins-Diehr, W. Wu and C. Youn, "The Problem Solving Environments of TeraGrid, Science Gateways, and the Intersection of the Two", Fourth IEEE International Conference on eScience, December 2008.

[42] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, S. Pamidighantam, “TeraGrid Science Gateways and Their Impact on Science”, IEEE Computer 41(11), 2008.

[43] R. Paul, “Compromising Twitter’s OAuth security system”, http://arstechnica.com/security/guides/2010/09/twitter-a-case-study-on-how-to-do-oauth-wrong.ars (September 2, 2010).

[44] B. Demeler, “UltraScan A Comprehensive Data Analysis Software Package for Analytical Ultracentrifugation Experiments”, Modern Analytical Ultracentrifugation: Techniques and Methods, D. J. Scott, S.E. Harding and A.J. Rowe. Eds., Royal Society of Chemistry (UK), 2005.

[45] UltraScan Reference and Application Database, http://www.ultrascan.uthscsa.edu/search-refs.html.

[46] XML Metadata Concept Catalog (XMC Cat), http://pti.iu.edu/d2i/xmccat