LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory / LIGO Scientific Collaboration

LIGO-T080058-00-U *LIGO* DRAFT March 7, 2008

Universal Computing Authentication and Authorization for the LIGO-Virgo Community

Authentication and Authorization Subcommittee of the LSC Computing Committee

S.Anderson, W.Anderson, D.Barker, K.Cannon, S.Finn,

S.Koranda, J.Minelli, T.Nash (Chair), S.Roddy, H.Williams

1 Introduction

When a person is granted access to a controlled computing resource two things must happen. The computer must have information to *authenticate* that whoever is knocking at its door is a known person (or at least a person associated with a known group) – just as we humans look through a peephole to see if we recognize who is ringing the doorbell (or at least if they are wearing a UPS uniform). If the person has been *authorized* to access the specific resource, the system permits this – like opening the door.

In the fluid and friendly LIGO environment, mechanisms for authenticating and authorizing grew up organically. System managers, and those who set up web sites and elogs and wikis, and all kinds of tools to support the collaboration, used whatever authentication/authorization approach was near at hand. Often this was a simple username/password combination. Often it was a broadly shared password meant to authorize all members of the LSC or a subgroup. Sometimes it was a complex government issued crypto-credential.

History

As early as 2006, the LSC Computing Committee (CompComm) became increasingly concerned by the wasted effort and the security issues resulting from this wildly non-uniform approach. It was no longer friendly, as members of the community needed to remember the complexities – and passwords - of a wide variety of access mechanisms (see Appendix II for examples). And, for system and collaboration tool managers, it was no longer convenient to have to maintain varying access control lists over a large number of systems. And it was no longer acceptably secure since the awkwardness and non-uniformity of the mechanisms encouraged insecure shortcuts like shared passwords and potentially vulnerable control of off-site access to critical interferometer system.

In January 2007, the CompComm became aware of the LSC Roster project, which the Directorate had commissioned the PSU group to develop in support of collaboration author lists, MOU membership lists, mailing lists, etc. The committee quickly realized that this could be the critical underpinning for they kind of uniform authentication/authorization approach it was thinking about. A few months later the subject came up again and a subcommittee was formed to investigate the idea and to develop goals and requirements for a possible project to move LIGO to a more uniform authentication and authorization environment.

The subcommittee, known as the LSC Authentication/Authorization Subcommittee (AuthComm), issued a report in Fall 2007.[1] Along with a discussion of the justification for moving to a new realm and of the feasibility of doing so, this report itemized goals and requirements. It also proposed that a week long "hackathon" be carried out to learn more about potential tools that could be used, particularly those coming from the Internet2 consortium.

The report and proposal was presented to the LSC Directorate at the end of November. The Directorate understood the need for a new approach in this area and that it should be closely integrated with the roster. The Directorate approved moving forward to the "hackathon" and noted that a detailed plan would need to be prepared for the project (See Appendix I).

[1] LIGO-T070286-02-U: *A New Authentication/Authorization Approach for LSC/LIGO*

The hackathon was carried out successfully in January 2008 resulting in confidence that we could address the goals and requirements in a timely way with reasonable effort. With encouragement from the CompComm and the Directorate, the AuthComm proceeded to write down the plan in this document outlining how this project could proceed. The present members of the AuthComm are listed as the authors on the title page of this plan.

The LIGO-Virgo Community

The formal agreement between LIGO and Virgo was developed during a period after the AuthComm had been formed and begun addressing the issues. When the report was presented in November, the Directorate noted the importance of making access to LIGO data comfortable for our new Virgo colleagues.

In developing this plan, we have taken this advice seriously and expect that the tools and approaches we are advocating will meet Virgo's needs both for their internal requirements (if they choose to apply them there) and for access to LIGO resources. No choices are final at this point; we expect to work closely with Virgo's experts as we move forward.

Obviously for Virgo members to be able to authenticate in the proposed new realm, their names and attributes and privileges will have to be included compatibly in a database. Whether Virgo members should be added into an expanded version of the present LSC roster or included via federation mechanisms in an updated version of the existing Virgo database is an important architectural decision that will have to be made. Both options are technically possible. There are obviously practical advantages and disadvantages either way. Federation can be implemented at the LDAP level as discussed in the technical sections that follow. We defer to the LSC and Virgo leadership for guidance as to how to proceed.

Anticipating incorporation of Virgo members into our authentication approaches, we refer throughout this document to the *LIGO-Virgo Community*, the LVC.

Motivation for Computing Access Controls

Following are the principle motivations for controlling access within the LVC:

Reserved Analysis. A scientifically defensible analysis of LVC data requires a deep understanding of the complexity of the detectors, their control loops, and noise sources. Because of a history of false detection announcements that affected the credibility of the field, our community is particularly concerned that an extensive and informed review process precede any announcement of the first detection of gravity waves. Control of access to LVC data and preliminary analysis results is motivated by these considerations.

Reviews and committees. Before discussing their comments and questions with individuals responsible for material under review, reviews and oversight committees need to be able to exchange considerations privately.

Administrative and personnel information. Privacy and procurement considerations dictate a certain level of controlled access to this kind of information.

Critical Systems. Interferometer controls need access protection to avoid damage to equipment, undocumented parameter changes, and down time.

Data Integrity. Data acquisition and storage systems need access protection to avoid inadvertent or malicious changes to interferometer data.

System Management. Root and other highly privileged access to compute and web servers, data repositories, and other common areas are limited to system managers for security and maintenance management reasons.

Note that there are no national security, classified data, safety, or intellectual property considerations in LVC computing.

Usability and Effort Level Goals

The AuthComm's earlier report listed goals and requirements. Important extracts follow:

Goals

A new authentication and authorization approach should provide access control appropriate to reasonable specific security requirements of individual LSC/LIGO computing resources. For users the access should be transparent, uniform, and easy. For system managers and security personnel the maintenance effort, including that required to clean up after break-ins and illicit software infestations, should be significantly reduced from the present level.

Our vision is a single sign-on to as many of LIGO services and systems as is reasonably possible. This implies access privilege "decorated" authentication, where the access attributes are preferably defined in the Roster data base but also in some cases might be just simple local system access control lists.

Requirements

• Minimal impact on existing infrastructure and resources. Less than one person-year total effort on existing computing resources required to integrate with new access regime, not including the effort required to develop and commission to the new mechanisms.

• Brief, accurate, up-to-date web based instructions maintained for all client and server components, readily comprehensible by all actors using those components.

• Significantly reduced system manager installation, maintenance, and user support effort in environments listed above.

• Authentication/authorization response time of 1 second or less (subject to intrinsic network latency).

• High availability: >5 minute non-response less than once per year; >1 minute non-response less than once per month.

Computing Activities and Actor Groupings

The principle areas of computing activity and how groups of individuals ("actors) relate to them within LIGO are shown in the figure below and itemized here:

Individual laptops and desktops

LIGO Lab desktops have been traditionally Windows or Solaris based. Control room workstations are Solaris. Scientists and many engineers work from laptops with Linux, NT/Vista, and Mac OSX, often dual boot. Workstation computers in the observatory control rooms are on the firewalled CDS network. At the observatory and LIGO Lab sites there is a broader General Computing (GC) network.

Analysis systems

Analysis is conducted in a grid environment (LIGO Data Grid, LDG), using Open Science Grid (OSG) tools, Linux. Each ~ year, the LSC CompComm designates a reference Linux platform to focus support and development (now RedHat with Debian in use at GEO and being considered as a secondary all-LIGO platform).

Einstein at Home

A SETI-like program for pulsar analysis of gravity wave data on volunteered, mostly external-to-LIGO home computers.

Web/Internet based information services

Web/internet based information is common with servers at dispersed sites including all the LIGO Lab and Tier 2 sites and some LSC universities. Information includes: CVS software repositories, document repositories, minutes and email logs, Wikis, electronic log books, interferometer status including quasi-real time analysis,...

CDS controls and data acquisition

Device control and data acquisition electronics includes embedded software managed from Solaris systems, data buffering and writing systems. Workstations on CDS network have MEDM controls screens and well as data monitoring tools. Access from the GC network and offsite goes through gateway computers at each site.

The main actor groups with special roles and access are:

LVC data analysis including Virgo

Includes all active LSC personnel and Virgo members (for certain data sets). Analysis is carried out within 4 working groups, which exchange information on archived mailing lists and via electronic logbooks. Working groups may limit access to their in-progress analysis products to their members. The working groups are: Inspiral, Pulsar, Burst, Stochastic

Upgrade projects

Enhanced and Advanced LIGO groups maintain wikis, schedules, archived mailing lists and function, much like an analysis working group.

Administrative (budget, purchasing and personnel)

Access to procurement and personnel files.

CDS development

Have access to CDS critical systems (controls and data acquisition), including off-site access through firewall(s).

Detector characterization tools development

Have access to buffered real-time output of CDS for development of detector characterization and data monitoring tools

Operators and Commissioning Teams

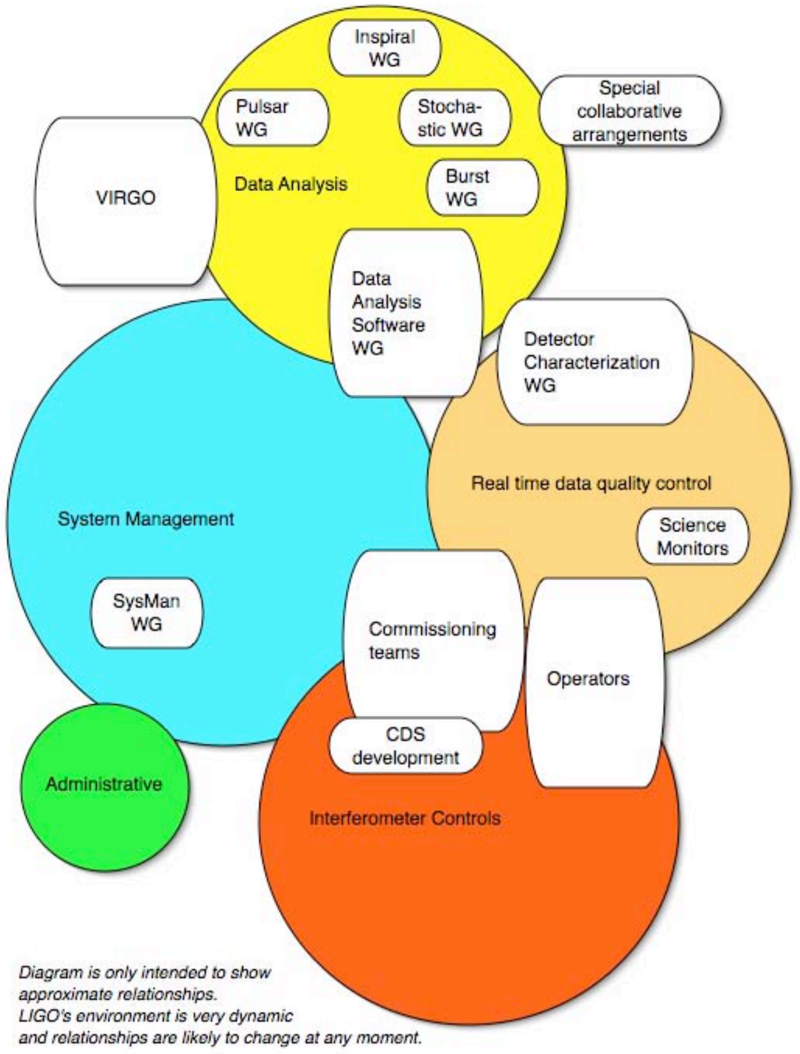
Access controls and monitoring consoles. May change control loop parameters, etc. Key commissioning personnel have off-site access.

Science Monitors

On shift "Sci-Mons" monitor control room workstation screens (and may use their laptops through the gateway) and access log books, wikis, data monitor tools, data characterization tools, but generally cannot change detector control parameters.

Special external collaborators

In order to support our computing activities, it is sometimes necessary to grant access as appropriate to external personnel, for example, for computing maintenance or where we have collaborations with external computing development groups.


Inspiral WG
Pulsar WG
Stocha-stic WG
Special collaborative arrangements
VIRGO
Data Analysis
Burst WG
Data Analysis Software WG
Detector Characterization WG
Real time data quality control
System Management
Science Monitors
SysMan WG
Commissioning teams
Operators
CDS development
Administrative
Interferometer Controls
Diagram is only intended to show approximate relationships.
LIGO's environment is very dynamic and relationships are likely to change at any moment.

2 Executive Summary

This document is long and full of technical reference material that has been researched by members of the AuthComm. In this section we provide the essentials to understand the proposed plan in terms of a view of the new world as might be seen by our members in their day-to-day activities. In the next section we outline the technical architecture that this plan proposes to support this friendly and efficient new access environment.

2.1 A new experience: Easy access to LVC Computing

The best way to introduce the proposed new authentication/authorization realm is from the perspective of a member of the LIGO Virgo Community accessing computing resources.

In the new world, each person will only have to remember a single "LVC username" and password to access all LVC computing resources[2]*.*

By LVC computing resources we mean the entire collection of services, information providers, and computational tools discussed in detail in other parts of this document.

By username we mean a single identifier, screen name, handle, or nickname that identifies a unique LVC person. The LVC username has associated with it a secret phrase or password that the user provides when required to authenticate or prove his identity.

There will be minimal restrictions on choice of a username. A few examples of LVC usernames might be scott.koranda, wanderson, nash, SamFinn, Stuart Anderson, HannahWilliams01, kip.

The mapping of the username or identity to the actual identity of the LVC person is a well known and public assertion. The associated secret phrase or password, however, is only known to the LVC person to whom the associated LVC username is mapped.

The mapping of a username to the identity of an LVC person is one-to-one. No two individuals share an LVC username.

Note that the LVC username may or may not be the same as a particular account name for a LIGO resource. So the LVC username 's.koranda' for the LIGO collaborator 'Scott Koranda' may or may not be the Linux account name that Scott Koranda is given for login access to a LIGO Data Grid cluster head node.

Managing LVC usernames and passwords will be easily accomplished via a web browser interface to the LVC Roster. The individual, or a PI or other manager, initially chooses the LVC username, typically when the person first joins the community. At any later time, however, a person may change their username. The LVC username chosen will have to meet liberal technical and policy guidelines (which will be detailed at a future time). We expect that each person will find enough flexibility in the technical and policy guidelines to easily find a LVC username that is comfortable and familiar.

[2] A few people who need access to other realms with special authentication requirements (e.g. Open Science Grid) or to internal areas with special security concerns (e.g. off-site access to CDS interferometer data acquisition and controls) may need an additional password (and possibly username) or a cryptocard.

Changing an LVC username and/or password will require browsing to an LVC Roster web page, authenticating with the current LVC username and password, and then changing the username and/or password.

LVC usernames will be restricted to avoid misleading others as to the actual identity to which an LVC username is mapped. For example we expect it will not be allowed that the LVC username 'scott.koranda' maps to the LIGO collaborator named Albert Lazzarini.

Restrictions on passwords or pass phrases will be stricter so that the password cannot be easily guessed or "cracked". Specific strategies and requirements on passwords and pass phrases will be detailed and enumerated at a later time, but we note that the main goal is to make the password "strong" and that we aim to support as many avenues as is reasonably possible to achieving a strong password. (Generally as the length of a pass phrase increases, the restrictions on characters and combinations of characters decreases, and each person will be allowed to chose their own balance subject to a criterion on password strength as measured by an automatic password checker.)

Forgetfulness will be supported. Memory challenged users may browse to a LVC Roster webpage, enter their LVC username, and request that their password be changed. The system will then email a reference to a unique session URL, valid for a limited time, where a new password may be entered (just like the current LIGO Roster mechanism for forgotten passwords). A new password become effective in a short amount of time after it has been successfully entered at the unique URL if it passes the strong password tester.

Our goal is single sign-on.

To the extent it is possible for the infrastructure to support it, an LVC person will be prompted only once per working session for an LVC username and password. After successfully authenticating, users will be "signed on" and able to access all LVC resources they are authorized to access, without having to again enter a username and password combination.

In the following we describe in detail the single sign-on experience for three separate classes of computing resources that require authentication and authorization for access: web page and wiki access via a web browser, LIGO Data Grid access using grid tools, and login access via consoles and shells.

Web access

At the beginning of her work day a LIGO collaborator, Gaby Gonzalez, starts up a web browser and attempts to load a protected web page or wiki page only available to LIGO-Virgo collaboration members; for example she may wish to visit the science documents page where drafts are posted for comment before being submitted for publication in journals or online archives.

Before the page loads, however, a dialog box appears and prompts her specifically for her LVC username and password. After correctly entering her LVC username 'gonzalez' and the associated password the web page is displayed and she can download and view a preprint of a publication up for review by collaboration members.

A short while later Gaby decides to visit a different protected LVC web page. For example, she may decide to visit the web pages hosting the wiki for the Compact Binary Coalescence (CBC) working group. Because she has already authenticated using her LVC username and password during this browser session, she is not prompted again for her username and password and is able to immediately access the web pages since Gaby is an active member of the CBC group.

Furthermore, when she decides to post a comment on the latest efficiency curves, she needs no further login to edit the wiki – the wiki has "learned" from the web infrastructure who she is and correctly attributes her comments automatically. Had it been a wiki to which she had only read access, it likewise would have denied her edit request, again without further prompting for her identity.

Still later Gaby is clicking through the web pages for the LIGO Data Grid and sees the link for the LIGO Computing Committee (CompComm) web pages. She is curious about what onerous burdens this nefarious group is planning to load on her shoulders, and she clicks on the link and tries to load the CompComm web pages. She is not prompted for her LVC username and password because the infrastructure already knows her username is 'gonzalez', but she is denied access to the CompComm web pages because she is not a member of the secretive CompComm and so is not authorized to view their web pages. She sees a simple page that explains she is not authorized to view the content.

Throughout the day Gaby continues to visit LVC protected web pages and wikis and is able to view pages for which she is authorized, but without having to enter her LVC username and password. At the end of her working day she quits her web browser, shuts down her laptop, and travels home.

Later that evening after a delicious early dinner, she starts her laptop and opens a new browser session. Gaby again attempts to load the wiki pages for the CBC group. Because this is a new browser session she is immediately prompted for her LVC username and password. After correctly entering 'gonzalez' and her password she is able to view the CBC pages.

Grid access

Alan Weinstein arrives at his office in the morning and decides to start a new inspiral workflow analysis on the cluster at UWM. Sitting down at his laptop he runs the command 'lvc-logon' and is prompted for his LVC username and password. Alan then uses 'ssh' to login to the cluster head node at UWM but he is not asked for either a login or a password again, but is automatically logged in since he is authorized to have access on the UWM cluster.

Alan then runs the necessary commands to generate a CBC inspiral analysis workflow and submit it for processing to the UWM Condor pool. Condor accepts his workflow because his LVC username and password were automatically carried with him in a credential when he used ssh to logon to the Milwaukee cluster, and that credential has automatically been made available to the Condor tools. The Condor daemons used the credentials to understand that it was Alan Weinstein who submitted the job and checked automatically to find that Alan is authorized to run jobs on the cluster.

A number of jobs in the inspiral workflow for finding segment lists and interferometer data also have access to the credentials that were automatically carried to the Milwaukee cluster for Alan and use them to authenticate on Alan's behalf to the services that answer requests for segment information and data locations.

Some of the jobs submitted on the Milwaukee cluster are picked up by the Condor tools and forwarded on that particular day to the Caltech cluster since the load at the time on the Caltech cluster happens to be lower. Again Alan's credentials are automatically and securely forwarded on his behalf so that the jobs that do run at Caltech can query and authenticate to the local segment and data services.

By default the credentials that Alan created when he ran 'lvc-logon' are good for 12 hours. Some jobs in the inspiral analysis, however, will run for 72 hours and will need access to credentials for Alan throughout the entire time so that they may authenticate to services on his behalf. To accomplish this at any given time during the workflow the current set of credentials can be used to authenticate to a credential repository and automatically obtain a fresh set of credentials good for another 12 hours. The credential management and refreshment happens without any intervention by Alan.

Later that day Alan is in his lab working on the 40M instrument. He has left his laptop in his office but decides while in the lab to login to the cluster at LHO from a lab machine and submit a second inspiral workflow. Because he has not run 'lvc-logon' from that lab machine recently he has no credentials there and so before using ssh he runs 'lvc-logon' and again enters his LVC username and password. He then uses ssh to login to the cluster at LHO and again his credentials are automatically forwarded with him so that he can submit the inspiral analysis workflow. When he is done submitting the workflow Alan logs off the LHO cluster. Since the lab machine is a shared machine and he does not want others to have access to his credentials he then runs 'ligo-logout' before he heads back to his office.

The next day Alan flies to a CBC face-to-face meeting only to find that he has forgotten the power cord for his laptop and his battery is dead. He borrows a laptop from another scientist at the meeting so that he can again submit another inspiral analysis workflow to a LDG cluster. Even though Alan has never used this computer before he is able to type 'lvc-logon', enter his username and password, and obtain the necessary credentials to again allow him to login to a remote cluster and submit the workflow for processing.

When Alan is done submitting the workflow he again runs 'ligo-logout' and hands the laptop back to its owner.

Alan continues to be able to run 'lvc-logon' and work in this way year after year--he never receives email about expiring X.509 certificates and he never again runs scripts requesting or renewing a certificate.

Login access

Fred Raab arrives at LHO in the morning and sits down at his Microsoft Windows workstation. After logging in and reading his email he decides he wants to look at some data files relating to last nights running. He clicks an icon in the system tray and opens up a dialog box into which he types his LVC username and password in order to create a short lived credential (a Kerberos ticket good for 12 hours by default). After obtaining the credential he then clicks on a terminal program and uses it to login remotely to blue.ligo-wa.caltech.edu, a UNIX computer, which is the gateway to CDS. He is not prompted for a login and password because the terminal program uses his Kerberos ticket credential for authentication to the UNIX workstation.

Later in the day Fred walks to the control room and sits down at a workstation on the CDS network, because he wants to check a DMT file he stored in his area from a test he ran the night before at 3 AM. He is prompted for a login and password and enters his LVC username and password and is logged into the machine. Day or night, no matter which UNIX workstation at LHO Fred wants to log into, or which LVC website, he always uses the same login and password--his LVC username and its associated password.

Our long-range goal is single sign-on across web, grid, and login spaces

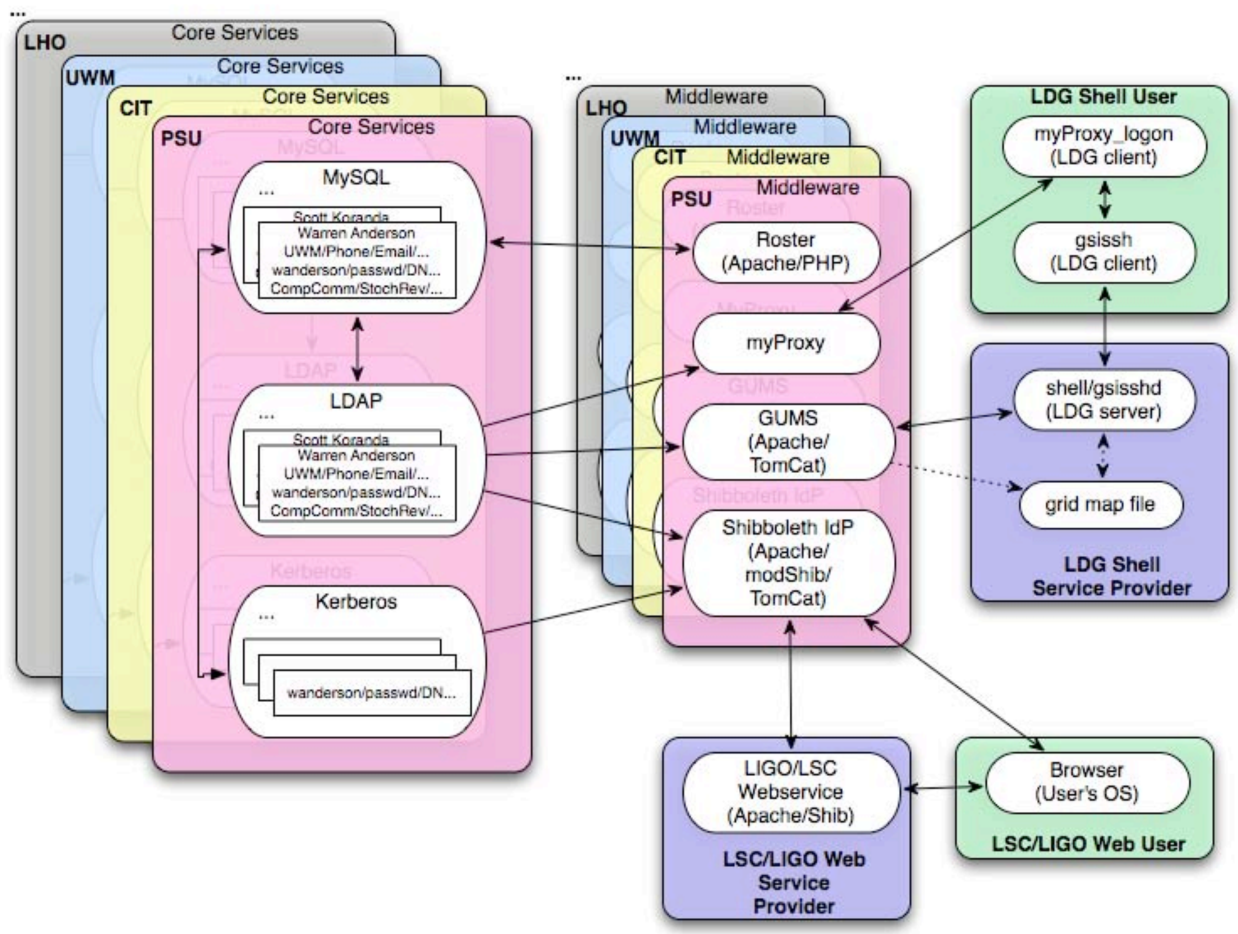
Throughout the scenarios above we have implicitly assumed no single sign-on that bridges directly across the web, grid, and login "spaces" or "realms". For example, after users sign on to the web space by browsing to a LIGO protected web page and entering when prompted their LVC username and password, they must still later run 'lvc-logon' when logging in to a remote Linux cluster head node on the LDG. They get to use the same LIGO username and password for each of these actions, but would explicitly have to take two actions - one to obtain single sign-on for the web space and one to obtain single sign-on for the grid space.

A technology does exist, however, that would allow a good fraction of LVC members to experience true single sign-on across these spaces. Using this client-side technology LVC collaborators could begin the day by running 'lvc-logon' once, and only once, to transparently obtain a suite of credentials or tokens. These credentials would then be available to all the tools they use from a single workstation for the rest of the day, including the web browser. That is, after running 'lvc-logon' and entering their LVC username and password they would be able to later start up a web browser and go directly to a protected web page or wiki without having to enter a username and password, and would be able to access grid services on the LDG, or login to remote workstations and the head nodes of Linux clusters.

This possibility is available to a large fraction of LVC people, but, depending on the combination of operating system and web browser, not to all at this time. We will educate the community during the rollout phase about compatibility issues like these.

3 An architecture that works auto-magically

In this section we lay out the key technical components of an architecture that allows use of a single but unshared credential in almost all cases and, to the extent feasible, also allows for a single sign-on each day to access many LVC services. A block diagram of the proposed architecture in its initial phase is shown in the figure.



Breakdown of services

For the purposes of this document, we consider the following breakdown of services in our community:

- **Web services** – these are services, which are usually accessed through a web browser. We focus here on services with access controls, since publicly viewable web services need no authentication or authorization. The services we consider include:
    - static web pages which need only basic authorization, such as http://www.ligo.caltech.edu/docs/ScienceDocs/

- enotebooks, inotebooks, elogs, such as http://www.ldas-sw.ligo.caltech.edu/ilog/pub/ilog.cgi?group=stochastic
- wikis such as http://www.lsc-group.phys.uwm.edu/ligovirgo/meetings/
- trouble ticketing services, such as http://www.lsc-group.phys.uwm.edu/cgi-bin/gnatsweb.pl?database=LSC-Data-Grid
- mailing list support (archives, etc), such as http://www.gravity.psu.edu/mailman/listinfo/ligosysadm
- the document control center
- the LVC roster

- **General Computing (GC) services** – these are LIGO Laboratory general computing services which are accessed by laboratory personnel and visiting scientists.[3] These include:
  - console login
  - ssh between systems
  - email (imap, pop, smtp) service

- **LIGO Data Grid (LDG) services –** these are services either adopted or developed by LIGO to access LIGO data and LDG. These tools currently
  - ssh access to LDG head nodes
  - grid job submission (globus-job-run, condor_submit, etc)
  - tools for accessing LIGO data (LSCdataFind, LSCsegFind, etc)
  - host-to-host communication for LIGO grid services (LDR, etc)

- **Other Services** – there are currently services, which do not readily fall under any of the above categories but are nonetheless critical to the operation of the LVC. These include:
  - version control systems (currently predominantly Concurrent Version System - CVS) for code repository, collaborative document development, collaborative web-page development, etc
  - restricted access mailing lists (currently predominantly mailman) which restrict subscription, posting, etc

- **Control and Diagnostic Systems (CDS) services --** these are services, which differ not in type but in need. CDS services primarily consist of web services and shell (usually command-line) access. Thus, from a technical standpoint, the needs are identical to those listed in for similar services above. However, CDS systems possess important special characteristics that require special attention:
  - CDS systems are critical to the operation of LIGO – CDS failures lead to profound if not catastrophic repercussions for LIGO science. These systems require robustness, a lack of complexity, and security against external attack.
  - CDS systems are intrinsically relatively secure –physical and network access to these systems is carefully managed.

[3] Other member institutions within the LVC are responsible for their own computing infrastructure. These institutions may choose to adopt our GC solution for their local collaborating group if they find it convenient.

- CDS systems require easy and quick operating and commissioning access to many workstations in the control room and interferometer areas.

Current Access Controls

Current access controls are described in Appendix I. They fall short in at least three areas:

- **Security** – Shared passwords are easy for users (especially in the case of a single collaboration-wide password for many services) but they are insecure – shared passwords must be transmitted to users, and are often shared between users through insecure channels. Individual passwords are little better if they cannot be set by users – complex access controls generally encourage users to circumvent or shortcut best practices (for example, keeping password lists in wallets, or refusing to change passwords without being required to, sending them through clear text channels, …).

- **Ease** – just the number of authentication mechanisms LVC people must keep track of is a source of frustration. Some of the mechanisms, such as certificates for LDG access, have been a continuing burden for a substantial number of users. The administrative load of administering so many types of credentials is at least equally burdensome for administrators, as is support for frustrated users.

- **Currency** – at present, there is no global mechanism for insuring that user credentials actually belong to current LVC members. Administrators of each service are responsible for the access controls for that service. Thus, new users must contact any number of administrators in order to gain access to services. When users leave the LVC, there is no impetus for anyone to notify all administrators to have access to services removed. Conversely, system administrators have not had easy access to comprehensive and current information about LVC membership in order to periodically remove users who should no longer have access. In the case of shared account access, there is not even the option to restrict access to only current members.

Proposed Access Contols

The problem of providing secure, easy and current access controls for a community like the LVC, which is geographically widely distributed, technically heterogeneous, and which has decentralized management, is not trivial. Nonetheless, almost all of the problems described above with current access controls are avoidable. They have grown organically because services have evolved responding to only local pressures, without global frameworks to unify and rationalize access. We propose to provide these global frameworks (to the extent it is feasible) for each of the service classes listed above. These frameworks are comprised of the following components:

- **Core Services**- the basis of all access controls for services offered within the community is membership management. Simply put, if we don't know who is in the LVC, there is no way to restrict access to services to only LVC members or subsets of them. The existing LIGO roster has taken a first step in this direction. By leveraging the LIGO roster, the membership of the majority of LVC service users can be managed. In order to leverage the same access controls for all LVC members, LIGO Lab staff, who are not in the roster (which presently includes only those listed in a LIGO MOU attachment Z) and our Virgo collaborators will need to be managed via an extension of the present roster, either by expanding its contents or supporting a parallel structure of data bases. We will discuss these options below.

The existing roster project currently provides two auxiliary services, which are central to our proposal:

- Kerberos is the current gold-standard for computer authentication services. Kerberos provides a very secure single sign-on (SSO) infrastructure for a wide variety of services. Users sign into a Kerberos system once per day using a password. Kerberos transparently provides the user a credential that can then be used to authenticate against other Kerberos-enabled service. Thus, after obtaining the Kerberos credential (known as a "ticket"), the user will be granted (or denied) access to any requested Kerberos-enabled services automatically, with no further action required.
- LDAP is a widely deployed directory service that interacts with a wide variety of applications and services that need information about users. For our purposes here, LDAP will be the source of information about authorization decisions. This is a considerably more complex task than the authentication functions served by Kerberos. Authorization decisions will be based on group membership attributes associated with each user. For example, users who have the Computing Committee group attribute in LDAP will be allowed to access services restricted to CompComm members, but other will not. Groups may be subsets or supersets of other groups, or may be defined by intersections, unions or complements of other groups. As with databases, forward-looking schema design is the key being able to leverage LDAP for as many different dependent services as possible. This is discussed further in a later section.

The Roster, Kerberos and LDAP services form the core services that allow membership management and allow us to translate membership information into authentication and authorization decisions for LIGO services.

- **Administrative Services** –
  - Grouper is a web-based Internet2 tool to create and manage group attributes within LDAP. Administrative privileges for group management are assigned via LDAP group attributes themselves. This allows each group attribute to have an associated administrator. Grouper could, for instance, allow the CBC review group leader to manage who has access to the CBC review group wiki. It would also allow the CBC review group leader to create a new subgroup of the group, for instance, for the review of primordial black hole binaries, if desired. Finally, it will allow group administrators to delegate administrative authority for groups under their administration. For example, the LIGO CBC review group leader could delegate administrative authority for the CBC review group attributes to the Virgo CBC review group leader and vice versa.

- Signet is a web-based Internet2 tool to manage privileges. We initially see privileges as being managed through group membership – ie group members have privileges associated with that group. However, it is possible to define privileges in more complicated ways than can easily be captured using group membership attributes. If this should prove necessary for users of our services, Signet will provide a simple mechanism for defining and managing privileges.

- **Web Services** –

  We propose to use Shibboleth, developed by Internet2 for distributed web services such as those on a multi-campus educational institution. Shibboleth provides a single sign-on (SSO) solution to users for accessing web services. There are two types of components for Shibboleth systems:
  - Identity providers (IdPs) provide identity information to web services about users trying to access them. We intend to use Kerberos as the source of authentication for the identity provider. The IdP will obtain group information from LDAP and provide it to web services that need to make group level access decisions. For an ideal Shibboleth implementation, users would be authenticated by their home institutions. This will not be feasible, at least initially, for us, since only a few LVC institutions are equipped to provide and identity for Shibboleth. Instead, we propose to have a single identity provider for LVC members controlled and maintained by our community. The crucial back-end service for this, the Kerberos service, is already in place thanks to the roster.
  - Service providers (SPs) provide web services to users. All SPs draw in the IdP for authentication and authorization information. SPs are therefore far more widely deployed in general than IdPs. Every web service in the LVC would have to be deployed as an ISP. Fortunately, the investment to do so is minimal – a single apache module needs to be deployed and configured for each SP. The module and the configuration to make it work with the LVC IdP will be provided as a simple package to be installed by those wanting to provide web services to our users.

  Shibboleth provides single sign-on capability for static pages and for Shibboleth enabled services – for up to 12 hours (or any other definable time period), unless the user closes the browser. For web services that require further authentication, but are not Shibboleth enabled (eg ilogs) users will typically need to provide further authentication. The particular implementation of the web services listed above currently used by LIGO are not Shibboleth enabled. However, there are implementations of some of these services which are Shibboleth enabled, and we believe a transtion to them will be relatively easy. These include:
  - Confluence, MediaWiki and Twiki wikis
  - Sympa mailing list management pages
  - Document management system Dspace

  Services for which there are currently no Shibboleth enabled implementations include:
  - enotebooks, inotebooks, elogs
  - trouble ticketing services
  - the LIGO roster

Initial investigations with Shibboleth-enabling a clone of the LIGO roster have proved encouraging. Provided that web services are written in a standard language, such as perl or php, the process of Shibboleth-enabling the services appears to be relatively straightforward. Further, it appears that Internet2 has some interest in producing a Shibboleth-enabled trouble ticket system, since this service is widely deployed in many contexts. Wikis appear to be replacing enotebooks, inotebooks, elogs, ilogs, etc in some contexts. However, ilogs are an intrinsic part of many LIGO activities, particularly in the control rooms. We believe that the effort would not be large to adapt the present LIGO ilog software to support Shibboleth and group membership access.

- **General Computing services** –

  These services are the "bread and butter" of Kerberos and LDAP. Thus, we expect to be able to leverage the core services to provide single sign-on in to general computing in a well understood and documented way. The plans for each service are:
  - Console logins will use Kerberos for authentication via the PAM (pluggable authentication module) framework. The PAM Kerberos module comes by default with a wide range of OSes including most *nixes. The PAM Kerberos module delegates authentication to a Kerberos service. Once authentication is successful, it is possible to retrieve unix account details (username, userid, groupid, home directory, shell, etc) from LDAP automatically provisioned accounts, although to do so appears to impose certain restrictions on participating sites. Alternatively, we can let each institution insert local posix account information into the LDAP, although this requires some mechanism to inform sysadmins that new users want access to the systems and then to have sysadmins add the appropriate information to their system.
  - ssh logins can use either Kerberos or grid certificates (more about that in the next section) to authenticate. Both can be enabled for a user with a single command, since a Kerberos credential (ticket) can be leveraged to get a grid certificate (or vice versa). Unix account information can again be drawn from LDAP.
  - Kerberos has become a standard authentication for mail user agents (MUAs, or mail clients) to authenticate for both download (imap, pop) and upload (smtp) of mail. This requires configuration of the mail service, which is usually straightforward, and configuration of the MUA to use this authentication method as opposed to others (like a mail password). Configuration of MUAs to authenticate with Kerberos is generally no more difficult than configuration for password authentication, which is what is generally deployed.

- **Grid services** –

  Grid services as they are currently implemented are inherently single sign-on by nature – grid credentials can be forwarded and delegated as needed. Our proposal will bring the important further benefit of coordination with the current membership roster. This can be done because sshd (the entry point into a grid domain) can be made to query an LDAP for user information via a service called GUMS, which has come out of the Open Science Grid. By leveraging GUMS and the roster LDAP, as soon as a user has a certificate that is entered into the GUMS, the user can have access to grid space without any further action (i.e. applying for accounts) on their part.

We further propose to have grid certificates generated for users automatically when they are confirmed as members in the roster system. This will allow a user to access LDG resources within minutes of being registered with the collaboration. The grid certificates will be stored in a myProxy service so that users need not manage them (a process which many users find difficult). However, users will be given the option to withdraw certificates from the service and manage them if they prefer.

This model precludes the use of certificates from the DOEGrids Certificate Authority (CA) since it violates their policy (which requires that the owner of a user certificate must be the only one who ever has read access to the certificate private key). We have fairly extensive experience with setting up CAs in LIGO, so we could leverage that experience to set up a CA for that purpose. Alternatively, the TeraGrid initiative has offered to let us leverage their myProxy service – we have not had time to explore this opportunity, but it might prove useful.

- **Other services** -
  Services in this category can leverage the Core Services outlined above to manage authentication:

  - Most version control systems admit a variety of authentication methods, including Kerberos. CVS, for instance, can be configured to accept external authentication methods. Currently, we use ssh as an authentication method in some cases. Since ssh can be made to accept grid credentials, in these cases we already have transparent controlled access for those with such credentials, and by utilizing the Roster Core Services to ensure ssh access only to current members, we will have overcome the last problem in maintaining CVS authentication. Alternatively, ssh can be made to use Kerberos for authentication, as outlined in the General Computing proposed implementation above. Finally, cvs has an internal gssapi authentication interface, although the option to compile this module is not always exercised in standard OS packages, so custom compilation might be necessary. The gssapi interface can leverage Kerberos for authentication without the need to delegate authentication to ssh. Which of these options we exercise will require further exploration. Likewise, there are a number of options available for using Kerberos to authenticate to SubVersion repositories.

  - Existing mailing list services can in many cases be modified to leverage membership information in an LDAP to determine who should be subscribed to a mailing list, for example by leveraging products such as the open-source LDAP Member Adaptor (LMA) product. Alternatively, Sympa is a Shibboleth enabled mailing list solution that would allow us to leverage the Core Services and Shibboleth to provide single sign-on and membership-aware mail services. Finally, the Roster project has implemented “mail exploder” services which are aware of current membership. Which of these options we wish to adopt for which current or future services again will require further study, but it is clear that this service can leverage core services for transparency and currency as well.

- **Control and Diagnostic Systems (CDS) services** The degree to which the suggested authentication and authorization infrastructure proposed above is appropriate within the CDS environment for those physically on-site is limited due to the nature and varieties of the platforms in this environment. Given the physical security, there does not seem to be strong justification for inserting authentication/authorization requirements into the path of commissioners and operators who need quick and shared access to consoles and workstations.

  However, given the above arguments for not locking down systems within the CDS domain and the critical nature of these services, restricting network access is certainly critical. This can be achieved allowing only Kerberos authenticated access to the CDS gateway. One immediate advantage is that along with strict control of who gets accounts, which is already done, global and central denial of authentication by revocation of the Kerberos principal becomes possible. This is a substantial improvement in our ability to deny access to a disgruntled employee to our most sensitive systems. Because of the problems with securing reusable passwords and remembering those that are strong enough and avoiding storage and transmission of passwords in the clear, it may be appropriate to use cryptocards with one-time passwords to authenticate at the CDS gateways. They can be made to work seamlessly with Kerberos.

Redundancy, replication, federation, availability, and all that

Because of our dependence on core services in this architecture, they must have high availability. In a widely distributed environment like ours, this implies that no LVC computing center should become locally unavailable due to the loss of an internet link to a center. This is addressed by having core services at each of the Tier 0,1, and 2 centers. At this time they are:– CIT, MIT, LLO, LHO, PSU and UWM. As other sites take on widespread computing support (eg Syracuse, GEO, Virgo) core services will also need to be supported there. Along with core services, other authentication services should be locally available, such as the Shibboleth IdP, a myProxy service and GUMS. In short, every site should have whatever services are needed to support access by their users.

Synchronization of the services will be through a master slave federation relationship, where one copy will be considered the master copy and other copies will synchronize with it. Service which draw information from core services will be configured to prefer the master copy but will fall through to other copies if the master is unavailable.

Human expertise will also need to be "federated" and "replicated" throughout our community so that a network outage cannot deprive us of critical human capabilities. For this reason, our plan calls for development of each component to be carried out in collaboration at two or more institutions.

4 Authentication/Authorization Component Modules

In this section we provide further details about the subsystem modules that make up the architecture just described. All the tools we plan to use are available to us from outside organizations, generally without modification and with significant commitments of expert support, or they are components that are already in use within LIGO and for which there is internal expertise. We will only use tools that are proven to be stable and robust. As we note below, some of the tools are in late stages of development and will be appropriate for our purposes when they have been sufficiently tested and are stable.

4.1 Internet2 tools

We are working closely with the Internet2, a collaboration of universities (http://www.internet2.edu/), which develops networking tools. We describe their tools first.

Internet2 (I2) is a consortium led by over 200 US universities working in partnership with industry and government to develop and deploy advanced network applications and technologies. As part of this mandate, I2 has developed Shibboleth middleware. When deployed, Shibboleth provides "single sign-on" (SSO) capability for users of web services - a user provide credentials (login name and password) at the first access to a web service provider which recognizes those credentials, but for the rest of the browser session (or until the configurable lifetime of the credential expires) the user can access other web services which require the same credential without re-authenticating - Shibboleth authenticates for them.

To augment this capability, I2 has developed associated middleware as well. The I2 web service tools in our proposed architecture are:

Shibboleth

As described above, Shibboleth provides single sign-on (SSO) capability for web services within a set of domains that agree to recognize common credentials. In the context of the LVC, these services could include static web pages, wikis, e-notebooks, trouble ticket systems, web-based code repositories, etc.

These web services fall into two categories, those that are Shibboleth enabled and those that aren’t. For those that are, not only is access to the service controlled by the Shibboleth credential, but the service can also draw upon these credentials to provide identification and administrative control. For instance, a Shibboleth enabled wiki could not only be accessed transparently via Shibboleth, but it would also know who accessed it, and be able to decide on that basis what wiki pages were readable and editable by the user, and would correctly attribute the edits to the user.

For services not enabled for Shibboleth, access to the service can be controlled by Shibboleth, but identity and access within the service (who can change which trouble tickets, for example) must be handled by some other (usually built-in) authorization infrastructure.

The mechanism by which Shibboleth operates is the following:

When a user first tries to access a Shibboleth protected web service, the service directs the user’s browser to a Shibboleth Identity Provider (IdP). The IdP authenticates the user through some external protocol (Kerberos, in this proposed plan) and then creates a token that contains the users identity and/or other attributes about the user (these attributes can be thought of as group membership information – what institute the user is from, which committees, etc). The information that is released is a fully configurable (restricted to the information the web service needs to know) subset of the information available about the user. The token with this information is then presented to the web service the user initially tried to access, and access control decisions are implemented by the web service provider based on them. On the next access to a web service that recognizes the same credentials, the user is again redirected to the IdP, and a new decision about which attributes to release is made. However, the identity of the user is now known, so there is no need to authenticate again. The new service provider can then decide whether the user has the correct attributes to access the service.

As an example of the implementation we envision for the LVC, we consider Jane Postdoc from the University of Minnesota. Jane, who works with the CBC group, comes into the office to find her background estimates done. She wants to post a graph of the results to the CBC wiki, which happens to be located at UWM. She opens her web browser and points it at the wiki. Since this is a new browser session, she has no Shibboleth token, so the CBC wiki redirects her to the LVC IdP, located at PSU. There, she uses her roster login name and password to authenticate via Kerberos against the Roster Kerberos Key Distribution Center (KDC)[4]. The IdP then queries the Roster LDAP to see what is known about Jane. Based on LIGO policy, a set of attributes pertaining to Jane are passed as a Shibboleth token to Jane's browser. Amongst them is the attribute that Jane is a card-carrying CBC member. Janes browser is now again automatically directed to the CBC wiki. The CBC wiki, which is configured to only let CBC members edit it, allows Jane to post her graph and correctly attributes the edit to her, because it is a Shibboleth enabled wiki.

Next, Jane is curious about a rumor she heard at the LV meeting last week that the CBC event found recently does not correspond to any blind injection. She decides to pull up the Detection Committee web page, which is hosted at MIT, to see if she can confirm this. She is not challenged for a username and password again because her identity is known to Shibboleth. However, her DetComm membership status might not be known (it probably wasn't needed for access to the CBC wiki, afterall). The IdP provides this new information to the new DetComm web page SP, and she is denied access to the DetComm web page, since only DetComm members are permitted to view it.

The burden incurred in implementing Shibboleth is relatively moderate. For the user, any modern browser with javascript and cookies enabled will work. For the web service provider, the web service must be provided by Apache and the Shibboleth Apache module must be downloaded and installed. The greatest work is done in configuring the Shibboleth IdP, where Java, the TomCat servlet, and the IdP Apache module must be installed and set up, Apache must be configured to access an authentication mechanism (like Kerberos), and the Shibboleth IdP must be configured to access an attribute source (like LDAP). For robustness, it is preferable that the authentication mechanism and the attribute source be local – thus, a KDC and LDAP instance should be maintained with the Shibboleth IdP. Fortunately, this need only be done at a single site, although we will want to do it at all Tier 0 through Tier 2 LVC sites to provide redundancy and failover.

[4] Other authentication possibilities exist, but we used Kerberos at the hackathon and were very pleased with it. Kerberos is described below.

Grouper and Signet

Since group membership is the central mechanism by which authorization decisions are made under Shibboleth, it is useful to have a simple interface to easily add and remove groups and administer their membership. I2 has provided such an interface in Grouper. Grouper provides a web interface to manage groups for authorized individuals. Authorization to manage groups can itself be authorized via Shibboleth. Group changes must be exported back to the attribute information service (i.e. LDAP) manually at this time.

Grouper is still in active development. Tests of this software at the hackathon led us to believe it may not yet be ready for production use. However, we believe it is likely that Grouper will be available, if not for the initial phase of rollout, then at a later phase.

Signet provides an alternate mechanism for authorization. Instead of having each service provider configure their service to authorize based on group attributes of the user in a configuration file, Signet provides a web interface that allows for access privileges to be set in a uniform and centralized manner. This software is in the early stages of development and was not tested at the hackathon, but we believe it would be worth taking a look at when it becomes more mature.

Comanage

Comanage is not so much a product as a packaging of products. It's purpose is to aggregate all the components of the core I2 middleware as well as some Shibboleth enabled collaboration tools into a single cohesive framework, allowing for ease of management and increased utility. Comanage was not ready to be evaluated at the hackathon.

4.2 Shibboleth-aware collaboration tools

There are a variety of web-based collaboration tools, which have been Shibboleth enabled, and more are being added all the time. The only one of these that we have directly evaluated is Confluence, a wiki, which is able to take advantage of the Shibboleth attribute information to automatically control read and write access based on user attributes. Confluence is a solid product, ready for production deployment. Other tools, which were not evaluated but look interesting are a telephony engine, an enterprise calendar system, a web meeting service, and a mailing list manager. These tools are discussed further in a later section.

4.3 Shibboleth-aware institutions

A significant number of universities and research institutions have shibboleth infrastructures. LVC members who are at these institutions will be able to use their local Shibboleth credentials. Unfortunately at this time, we believe this includes less than 20% of the LVC. Shib-aware LVC institutions include the following members of "InCommon" (see the I2 website): Columbia University, Northwester University, Penn State University, Stanford University, University of Maryland, University of Massachusetts Amherst, University of Rochester.

4.4 The Open Science Grid Authentication Tool - GUMS

Open Science Grid, OSG, (http://www.opensciencegrid.org/), is a DOE-NSF funded activity which develops grid support tools, which are extensively used to support the LIGO Data Grid (LDG). Of particular use for authentication is GUMS.[5]

Users authenticate to services within the LIGO Data Grid (LDG) using X.509 certificates and associated limited lifetime RFC 3820 proxy certificates. The subject or distinguished name (DN) on the certificate uniquely identifies each LDG user. An example DN is

/DC=org/DC=LIGO/OU=People/CN=Patrick Brady

LDG services authenticate users or client tools and services acting on behalf of users by requiring that they present an X.509 or proxy certificate properly signed by a recognized certificate authority (CA) and by requiring that the owners of credentials successfully prove they control the associated private key.

LDG services authorize users by comparing the DN on the presented credential to an access control list or mechanism. Currently LDG services use the simplest access control mechanism. Each service is configured to compare the DN to a static list of authorized DNs and will only authorize access to the service if the presented DN is found in the static list of authorized DNs. If the static list is kept in a separate file on the file system it is usually called a grid-mapfile.

Some services need to map from an authorized DN to a local account name. For these services the grid-mapfile contains a mapping from a DN to one or more local account names. A typical grid-mapfile has entries like this:

"/DC=org/DC=LIGO/OU=People/CN=Patrick Brady" patrick

"/DC=org/DC=LIGO/OU=People/CN=Scott Koranda" skoranda,root

"/DC=org/DC=LIGO/OU=People/CN=Warren Anderson" wanderson

"/DC=org/DC=LIGO/OU=People/CN=Sam Finn" lsf,root

"/DC=org/DC=LIGO/OU=People/CN=Alan Weinstein" ajw

LDG services that use grid-mapfiles for authorization include GSI-enabled sshd for remote logins, the Globus GridFTP server for remote file access, the Globus jobmanager and gatekeeper for remote job submission, the LSCsegFind server for serving information on interferometer GPS time segments, and the LDRdataFindServer for serving information on the location of interferometer data or frame files.

Currently each instance of each of these services across the LDG requires a separate static grid-mapfile for authorization. While static grid-mapfiles are simple to manage because they are plain text files, this mechanism does not scale across the entire LDG. The management burden is high for so many distributed and distinct grid-mapfiles, and removing authorization for any individual LDG user requires direct administrator access.

To move beyond static grid-mapfiles for LDG services we propose using GUMS (Grid User Management System).

[5] https://www.racf.bnl.gov/Facility/GUMS/1.2/index.html

GUMS is a Grid Identity Mapping Service. The GUMS server maps subject DNs to other user attributes such as local user account names upon request and returns the mapping to a grid service. GUMS is comprised of web services, web pages for GUMS administration, and command-line tools which interact with the web services. Typically, the term "GUMS" refers to theserver portion.

GUMS is particularly well suited to a heterogeneous environment with multiple grid services; it allows the implementation of a single site-wide usage policy, thereby providing better control and security for access to the site's grid resources. Using GUMS, individual resource administrators are able to assign different mapping policies to different groups of users and define groups of hosts on which the mappings will be used. GUMS was designed to integrate with the site's local information services. In particular GUMS can integrate with LDAP (see LDAP section below).

GUMS can be configured to generate static grid-mapfiles or to map users dynamically as each grid service is queried for access. If configured to generate a grid-mapfile, GUMS downloads the file to each service as scheduled or requested by an administrator via the GUMS client tools. If configured to map users dynamically and individually, GUMS is called by the service upon each service request by a client tool.

The advantages and disadvantages inherent in these two approaches to using GUMS (scheduled download and dynamic access on client request) must be balanced. There are intrinsic reliability concerns with any approach, like dynamic GUMS access, dependent on long range network accesses. These may be mitigated to some extent by a fail-over to a locally maintained static file. We are inclined to err on the side of robustness, and expect to start with locally maintained files updated on a frequent schedule. Our goal is that local copies are within minutes of being timely. This latency has important security related implications regarding the ability to rapidly withdraw access to a compromised identity name.

GUMS runs at a grid site under the control of site administrators; it is a "site tool" as opposed to a virtual organization, "VO tool". The mappings in a site's GUMS installation are defined in a single XML policy file. This file may contain multiple policies, and the administrator can assign different policies to different groups of users. The administrator can also specify different mappings on different hosts or different sets of hosts.

GUMS is designed to be extensible so that it can address specific site requirements. All the GUMS policy components (i.e., user group definition, mapping policies, and so on) are implemented via a few simple interfaces which can be implemented with almost no dependencies on GUMS. Thus a site administrator with very little knowledge of GUMS itself can add external code to manipulate GUMS functionality or data, e.g, to tell GUMS how to map credentials, or to pull GUMS data into a local storage system.

4.5 MyProxy

Users authenticate to services within the LIGO Data Grid (LDG) using X.509 certificates and associated limited lifetime RFC 3820 proxy certificates. The subject or distinguished name (DN) on the certificateuniquely identifies each LDG user. An example DN is

/DC=org/DC=LIGO/OU=People/CN=Patrick Brady

LDG services authenticate users or client tools and services acting on a user's behalf by requiring that they present an X.509 or proxy certificate properly signed by a recognized certificate authority

(CA) and by requiring that owners of credentials successfully prove they control the associated private key.

Currently a LDG user's X.509 certificate and the associated private key are kept on a file system as two small plain text files. Together the certificate and key are known as a credential. LDG user acquires a credential by running a script on their local machine to generate a private key and matching certificate request. The certificate request is uploaded by the script to a central file server.

An administrator or agent with appropriate authorization then verifies the veracity of the request to make sure that the DN on the request matches the actual identity of the person who submitted the request. Once verified the agent causes the certificate request to be digitally signed using the root certificate from the CA and hence transformed into a proper X.509 digital certificate.

A notice is sent to the user who runs another script to download the signed certificate and placeit into the proper location on the local file system. The script also attempts to make sure that the certificate and key files have the correct ownership and mode on the file system.

Today LDG certificates for people located in the U.S. are signed by the DOEGrids CA and are good for one year. Each time users' certificates are about to expire they must run another script to renew the credential. If a user's certificate has already expired, a new request is required and again must be verified by the CA agent or administrator. LDG users located in other countries most often acquire credentials from a national or regional CA using a similar routine.

We stress that today each LDG user is directly responsible for acquiring and managing their own X.509 digital credentials and that this burden is cumbersome for a large majority of LDG users.

MyProxy[6] is software for managing X.509 credentials (certificates and private keys). MyProxy functions as an online credential repository to allow users to securely obtain both proxy credentials and the original or "end entity" credential when and where needed. Users (or other tools on behalf of a user) run myproxy-logon to authenticate to the MyProxy repository and obtain credentials.

Storing credentials in a MyProxy repository allows users to easily obtain RFC 3820 proxy credentials, without worrying about managing private key and certificate files. They can use MyProxy to delegate credentials to services acting on their behalf (like a workflow management system) by storing credentials in the MyProxy repository. They can also use MyProxy to renew credentials so that, for example, long-running jobs don't fail because of expired credentials.

A professionally managed MyProxy server can provide a more secure storage location for private keys than typical end-user systems. MyProxy can be configured to encrypt all private keys in the repository with user-chosen passphrases, with server-enforced policies for passphrase quality. By using a proxy credential delegation protocol, MyProxy allows users to obtain proxy credentials when needed without ever transferring private keys over the network.

MyProxy provides a set of flexible authentication and authorization mechanisms for controlling access to credentials. Server-wide policies allow the MyProxy administrator to control how credentials may be used. Per-credential policies provide additional controls for credential owners. MyProxy supports multiple authentication mechanisms, including passphrase, certificate, Kerberos, Pubcookie, VOMS, PAM, LDAP, SASL and One Time Passwords (OTP).

[6] http://grid.ncsa.uiuc.edu/myproxy/

Via PAM, MyProxy can be configured to use an external authentication mechanism such as Kerberos or a remote LDAP server, instead of using credential encryption passphrases to verify identity.

At this time, Department of Energy policies preclude use of MyProxy for DOE Grid certificates. For this reason we will use an alternative Certificate Authority for LDG certs, as discussed below. OSG users of resources external to LIGO will still be required to use DOEGrids certs and self-manage them.

4.6 Kerberos

Kerberos is the name of a network authentication protocol, and also the name of a (free) implementation of that protocol published by MIT. Kerberos allows individuals communicating on non-secure networks to prove their identity to each other in a secure manner. Kerberos can also provide strong encryption of communications between users who have proven their identities to each other. Kerberos (both the protocol and the implementation) is currently at version 5, which has been stable since 1993.

Kerberos is used by Microsoft as their default identification method for Windows 2000, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008. Apple's Mac OS X uses Kerberos in both its client and server versions. Kerberos is also available in all modern Linux distributions. Organizations using Kerberos at the enterprise level include universities (e.g., MIT, Stanford, Penn State, Duke, Cornell, Carnegie Mellon, University of Michigan,...), businesses (e.g., Google, Apple, Sun, Microsoft) national laboratories (e.g., FNAL), governmental agencies (e.g., DoD, NASA), and international organizations (e.g., CERN).

The Kerberos protocol is managed by the usual RFC (Request for Comments) mechanism of the IETF (Internet Engineering Task Force). For this purpose the IETF has established a dedicated Kerberos Working Group (krb-wg). Kerberos is also supported by The MIT Kerberos Consortium, whose goal is to develop interoperable technologies to enable organizations and federated realms of organizations to use Kerberos as the single sign-on solution for access to all applications and services.

Kerberos and Shibboleth complement each other. Shibboleth is a standards-based, open source middleware software that provides Web Single Sign On (SSO) across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner. In our proposed architecture, Shibboleth will provide the Web SSO infrastructure and Kerberos will provide the ability for users to securely prove their identity to each other and to LVC service providers.

Kerberos and PKI (X.509) authentication also complement each other. Both involve a ``secrets keeper'' (a Certificate Authority - CA - in the PKI universe, a Key Distribution Center - KDC - in the Kerberos world), which is responsible for attesting to the identity of a user.

In the Kerberos model all users and services trust the KDC and the KDC facilitates trusted communications between users/services. The Kerberos model is ideally suited for establishing trusted communication between members or services within the same organization (who all trust the organization's KDCs).

The PKI system addresses the problem of establishing trusted communication between members or services of different organizations. The policy infrastructure, requirements and software implementation required to support PKI authentication is substantially greater because it must support trust communication between members and services of different organizations that have agreed to share certain services but otherwise do not generally trust each other. Since they address two different aspects of the same problem - trust within an organization and trust between organizations - the integration of Kerberos and PKI authentication is well-developed.

In the architecture proposed here, Kerberos will provide authentication within the LVC for web and shell (terminal) access, and PKI (implemented using X.509 certificates) will provide authentication in the grid environment.

4.7 Certificate Authority

Public key infrastructure X.509 certificates (hereafter just called certificates in this section) are the mechanism by which authentication takes place on the LIGO Data Grid (LDG). Certificates are a widely accepted standard for authentication of services and users. Their usage on the LDG was mandated by the fact that they have become the de facto standard for the grid middleware (globus, condor, etc) that we use.

LIGO currently relies on the DOEGrids, UK E-science and GermanGrid Certificate Authorities (CAs) to provide certificates. Our Virgo partners rely on the GRID-FR, INFN and DutchGrid CAs for certificates. There are a number of issues that are created by using certificates from these certificate authorities:

- **Limit of scope** – it would be much easier to support users if we were dealing with only one CA. However, grid CA’s are currently specific to a region – e.g. DOEGrids CA will only issue certificates to users at institutions within the USA. As well as increasing the support burden (we need experts in three or more CA procedures) it increases the user burden and creates loss of science opportunity when scientists (especially grad students and post-docs) move from one country to another. This is a significant burden on sysadmins as well, who have to change the users accounts throughout the LDG to be accessed by her new certificate rather than the old.

- **Strict enforcement of protocols** – For historical reasons, associated with extremely heightened perceived security concerns within the DOE during the 1990s, DOEGrids has very strict protocols and there is a strong agency reluctance to change them. For instance, only a user may have access to the private key that unlocks their certificate – if there is ever any reason to suspect that any other person (friend, sysadmin, computer repair person) the certificate must be revoked. This prevents user support personnel from directly troubleshooting certificate problems. Also, there is a strict upper limit on duration of a certificate of one year – users must renew certificates every year – failure to do so requires the user to gain a certificate with a different Common Name (CN, certificate jargon for user name), which then requires work on the behalf of sysadmins to have the new CN recognized on the LDG.

- **Generality of infrastructure** – Grid CAs like DOEGrids CA are used by a wide variety of organizations as a source of certificates. As such, they naturally wish to provide the most general and simple infrastructure for users to obtain a certificate. Unfortunately, simplicity and generality create a more difficult user experience. The DOEGrids method for delivering certificates provides certificates in the wrong form (PKCS12 provided, PEM required) for LDG usage. The delivery mechanism also leaves the certificate embedded in the user's browser, from which it must be exported. The LSC has implemented a number of scripts to make requesting, retrieving and renewing a certificate easier for users, but this has led to difficulties of its own. DOEGrids will not provide instructions to users based on the organization they belong to. This means that users have in the past received conflicting instructions – the script-based instructions from LIGO web pages and the instructions emailed by DOEGrids to users on how to perform actions with their general infrastructure.

Recently, the LSC, the Open Science Grid and DOEGrids tried to solve this problem. The success was mixed – LIGO had to change its infrastructure to use generic script names and the DOEGrids instituted a mechanism by which, once a certificate was either requested or renewed by script, a flag was set that indicated that the user wanted script-based instructions. Because a script needs to be used first, there is up to a one-year period where instructions will still be conflicting for some users. Also, the change in script names is causing some confusion amongst our user community. Another side-effect of the double infrastructure is that DOEGrids CA is concerned only with maintaining compatibility with their user interface – a recent upgrade of their software "broke" our scripts, which led to about a month of work and confusion for our users.

In general terms, our user and administrators have had difficulty in trying to deal with with Grid CAs in general, and DOEGrids CA in particular. Although some progress has been made in bridging the gap between their service and our needs, it has been slow and costly. Furthermore, fundamentally, the LDG does not require the strict adherence to security protocols enforced upon us by Grid CAs.

It is the consensus of the LSC Authentication Committee that users should not have to manage certificates from the Grids CAs. We recommend a centralized management of grid certificates with users responsible only for ensuring they have generated a valid proxy certificate (currently done with `grid-proxy-init` command). Since it is not possible to have centralized certificate management with Grid CA certificates, our proposed approach is to change CAs. The simplest solution may well be to simply have our own CA – Warren Anderson, Scott Koranda and Shannon Roddy are all members of the LSC with experience in setting up a CA. The burden of doing so, especially if users will not be directly accessing the CA, is minimal. The cost of doing so is that the certificates and their proxies will not be recognized outside the LIGO data grid – those who wish to use certificates outside the LDG (for instance, on the OSG) will need to have a second certificate (probably from DOEGrids CA) to do so. We believe that the set of users for whom this is an issue is small and expert enough that this will not be a problem for them.

Alternatively, TeraGrids has offered to manage user certificates and allow our users to draw proxy certificates from them. This has the benefit that these credentials are recognized throughout the grid world, so no user would have to maintain a second set of credentials for LIGO work. We would, however, again be tying ourselves to an external provider, with their own sets of protocols and interfaces. Fortunately, initial indications are that TeraGrids would be considerably easier to bridge the service/need gap with.

We will further investigate the best approach to having an acceptable CA, considering both having our own CA and utilizing TeraGrid's.

4.8 LDAP

In order to provide identity information to applications, one must store identity information in some form of database. To provide authentication services, this also requires some sort of username/challenge process. This can be as simple as local user accounts and passwords (e.g. /etc/passwd) or as complex as identity services which include everything from username/password pairs to X.509 public certificates, contact information, identification photographs, etc.

The most commonly used centralized directory service in use today is LDAP.[7] LDAP provides a standard means for storing various entries including identity information in a centralized and sometimes replicated directory service.

4.8.1 LDAP History

X.500

LDAP (Lightweight Directory Access Protocol) has its origins in the X.500 standards published in 1990. X.500 was one of the first general-purpose directory systems. X.500 or DAP (Directory Access Protocol, the DAP in LDAP) is a heavyweight directory protocol, which suffered from a number of problems.

DIXIE & DAS, then LDAP

During the 90s two groups tried to develop middleware to ease the burden between the desktop client and X.500 directories. The two implementations were DIXIE and DAS. (RFCs 1202 and 1249) Again, there were flaws and barriers in both of these implementations. However, the value of a lighter-weight directory access protocol was realized. Work later began on LDAP as first defined in RFC 1487 in 1993 and LDAPv2 in RFC 1777.

LDAP as a standard specifies the protocols, not the server implementation. Initially LDAP was itself just a piece of middleware between the client and an X.500 directory. LDAP later became a complete lightweight replacement for X.500 in around '95-'96.

LDAP became the Internet directory service of choice with the backing of the then big players on the Internet (Netscape, Sun, etc). Today, LDAP is "the" directory standard.

4.8.2 LDAP Server implementations

There are a number of vendors who offer LDAP implementations. Sun, Redhat, Novell, OpenLDAP, Computer Associates all offer server software. Even Microsoft's Active Directory is based in part on LDAP. We have not yet selected a particular server implementation. The plan is to determine requirements and make a selection based on requirements.

---

[7] Useful references on LDAP are:
donnelly, m. (2000). Retrieved from ldapman.org: http://www.ldapman.org/articles/tree_design.html#planning; and Timothy A. Howes, P. M. (2003). *Understanding and Deploying LDAP Directory Services*, Boston MA: Addison Wesley.

Since LDAP describes the protocol and is a strict standard, selection of the backend LDAP server should not affect the ability of Apache or other software to communicate with the directory server. All vendor LDAP implementations should interoperate based on the assumption that they comply with the published LDAP standard.

4.8.3 Integration with authentication mechanisms

Kerberos: Evident from documentation on the web, it should be possible to integrate LDAP and Kerberos for shell based authentication on various UNIX operating systems. Kerberos would be used for authentication while LDAP would be used for authorization, group identification, etc.

Best practice seems to be not to use LDAP for authentication, rather only for authorization and directory services.

Apache & web services: Through mod_auth_ldap (Apache 2.0) and mod_authnz_ldap (Apache 2.2), apache can authenticate directly to LDAP directories. There also appears to be a number of (third party?) modules for apache 1.3, however our feeling is that most LVC sites should already be migrated to Apache 2.x.

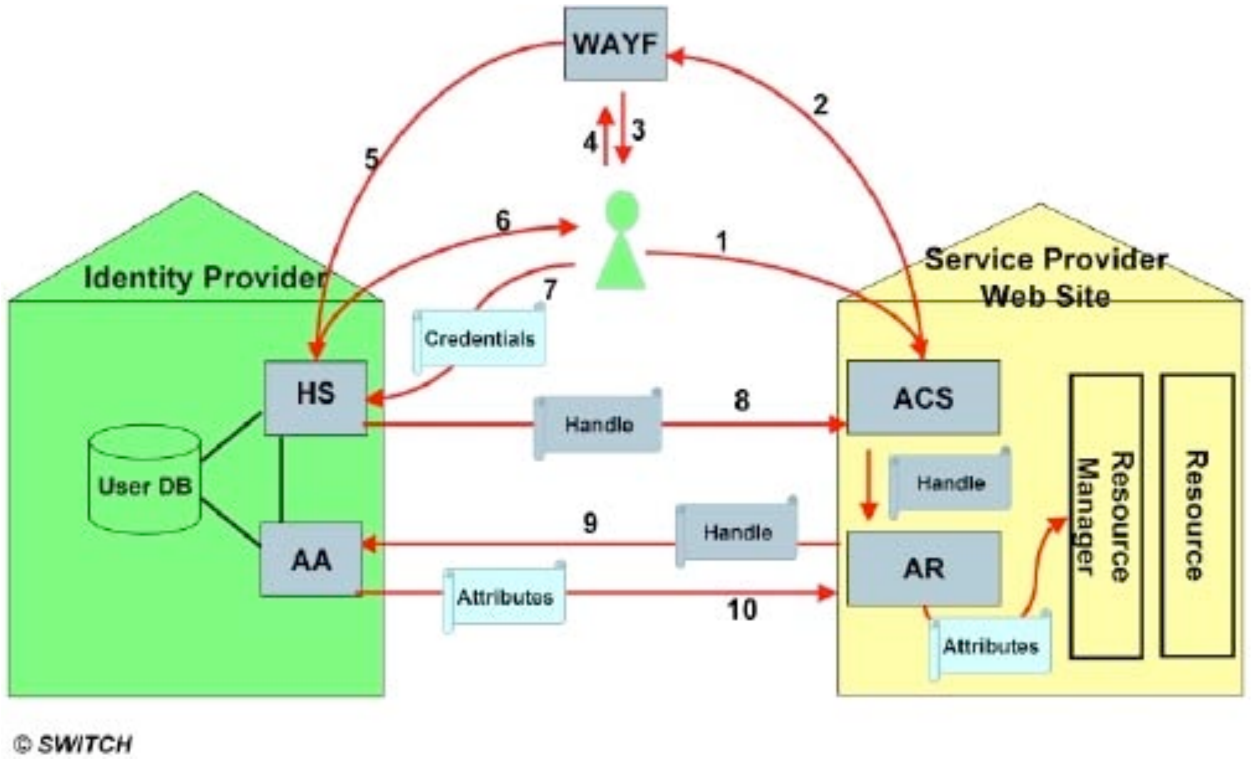
4.8.4 Integration with shibboleth

Copied from the Internet2 Shibboleth wiki (Internet2 wiki):

> *LDAP Groups Integration*
>
> *The restriction of access to a resource to a specific set of users can be implemented in a variety of ways. Traditionally, this has been done with standard LDAP groups, and there may be a need or desire to continue using these even as attribute-based groups are supported. There are several ways this integration can be performed. In a situation where there is no need to integrate with LDAP, there is a small number of users, or the users are scattered amongst a large number of IdPs, it's recommended that an AuthGroupFile be used.*
>
> *There are two ways integration can be achieved:*
>
> *1. Import the group information using the IdP, transport it using an appropriate attribute name, and export it as memberof using the following as an example:*
>
> <example removed>
>
> *2. Alternatively, assuming a few limitations on how the LDAP module works (specifically, it's at least implemented with r->user), it's possible to actually use the LDAP module itself (or any other auth/z module) for the auth/z and access control once Shibboleth transports the information. Everything from Shibboleth in the above example remains, but not the Apache AuthType or Require statements. These change based on the implementation of the auth/z module in question.*

In other words, as we understand it, Shibboleth can either directly query LDAP for group membership at the Identity Provider level, or it can rely on the Apache module to query for group membership.

In the rather complicated illustration below taken from the shibboleth website[8], LDAP can and often does serve as the User DB contained within the Identity Provider block. LDAP is not required, but appears to be the most commonly used backend.



4.8.5 Considerations for LVC implementations

Directory tree design

LDAP directories are hierarchical or tree based structures. There is much flexibility in the initial structure of the namespace. Each entry has one parent and any number of children. The base DN (Distinguished Name) of directory trees are generally based on domain names by convention though this is not mandatory[9]. An example root would be dc=ligo,dc=org. dc stands for Directory Component.

Below the base DN can be any number of entries, organizational units, etc. For example, there can be organizational units (OUs) based on geographic location, relationship to the top level organization, etc. So, the top level organization could be seen as LVC, with several organizational units below it, such as LSC, Virgo, etc. Below those levels entries could further be separated by OUs based on status in the organization, such as staff, postdoc, guest, etc.

Of course, there could be one flat structure and organizational data can be captured as an attribute of the entry. Currently, the LIGO roster is one flat directory consisting of a base DN of

[8] http://shibboleth.internet2.edu/tech-intro.html

[9] One exception is if integration with MS Active Directory will ever be desired. If so, the domain name part is mandatory. However, since we have an organization that spans multiple domains, this means other complexity besides just the LDAP tree.

dc=ligo,dc=org with individual directory entries describing members of the LSC just below the DN. The “o:” attribute (organization) is currently used to show organizational membership in the Roster LDAP. “o” can be multivalued, in other words there can be multiple entries for organization, such as LSC, CompComm, etc.

A generic LDIF would look something like this[10]:

```
dn: dc=lvc,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
o: LSC
dc: ligo

dn: uid=1.1,dc=ligo,dc=org
givenName: Joe
sn: User
mail: joe.user@ligo.org
o: Some University
uid: 1.1
cn: Joe User
telephoneNumber: 555 555 1212
facsimileTelephoneNumber: 555 555 1212
street: 555 some road
l: Somewhereville
postalCode: 11111
st: Somestate
objectClass: inetOrgPerson
objectClass: top
```

The following wise suggestions on directory tree design are from an internet posting (donnelly, 2000):

- If at all possible, avoid a directory tree design that will have to change. Try to stay away from directory trees based on a company org chart, a current business model, and the like.

- You'll want to avoid moving LDAP records from one OU to another. If you forsee this happening often with your current design, try to redesign. Can you consolidate two or more branches into one location? Can you use an attribute - location, department - to accomplish the same goal?

- Do you have types of data that are similar in some respects, but that differ widely in how the data is used? Can you create a separate OU for each *type* of entry? If entries will not

[10] Based on an LDIF taken from the existing LIGO roster LDAP

need to move from one OU to the other for any reason whatsoever, keep the groups separated. (Consider the example above, with ou=Employees and ou=ITaccounts.)

- If you'll be replicating data at some point, consider whether you can break an OU into sub-OUs based on the needs of your users. Think twice if doing so will result in an object moving from one OU to another. (In the example above, the OU for customers was a natural candidate for sub-grouping; the OU for employees was not.)

- If you need to hide a portion of the data in your directory, either for security reasons or simply to avoid confusion, you *may* want to use an OU for the task. If it means violating any of the guidelines above, you'll have to judge for yourself what to do - security is a tricky subject, and reasonable people can and do differ.

Decisions on the layout of the directory information tree (DIT) can have unforeseen results at a later time.

For example, appropriate choices can make implementation of links to LDAP easier in various contexts and thereby increase acceptance by system managers. In regard to one important context, Appendix II shows the default layout in LDIF format for a “Solaris Native LDAP” directory serving as a replacement for NIS+.

Clearly the design of the namespace will need to be carefully planned. We discuss this further in the section on Directory Services.

Schemas

The LDAP schema is the set of rules that defines what types of entries and attributes can be contained in an LDAP directory. LDAP servers generally come with a reasonable default schema and at minimum the set of schemas defined by RFCs. Schemas can be extended. The method in which the schema is actually extended may differ based on LDAP server implementation.

The minimum LDAP schema is defined in RFC 2252 and RFC 2256. The default schema that comes with the Sun LDAP server can be found via the online documentation[11]. Note that the Sun directory server (at least version 5.2) does not include the eduPerson objectclass. However, as is part of the LDAP design, it can be extended to support it. The default schema for OpenLDAP[12] and the fedora directory server[13] can also be found online.

Object classes and attributes

An object class is a set of rules that defines what attributes an entry may contain.

---

[11] http://docs.sun.com/source/816-6699-10/objclass.html#23738

[12] http://www.openldap.org/doc/admin24/schema.html

[13] http://www.redhat.com/docs/manuals/dir-server/schema/contents.htm

Each LDAP entry is assigned one or more object classes. The object class may have a set of possible attributes. Each attribute has a data type (text, binary, etc.) and a set of rules concerning the attributes such as min/maximum size, required or optional, etc.

For example, an inetOrgPerson[14] object class may contain the following attributes. Each attribute can contain information, which is defined in the schema.

audio

businessCategory

carLicense

departmentNumber

displayName

employeeType

employeeNumber

givenName

homePhone

homePostalAddress

initials

jpegPhoto

labeledUri

manager

mobile

pager

photo

preferredLanguage

mail

o

roomNumber

secretary

uid

x500UniqueIdentifier

userCertificate

userSMIMECertificate

userPKCS12

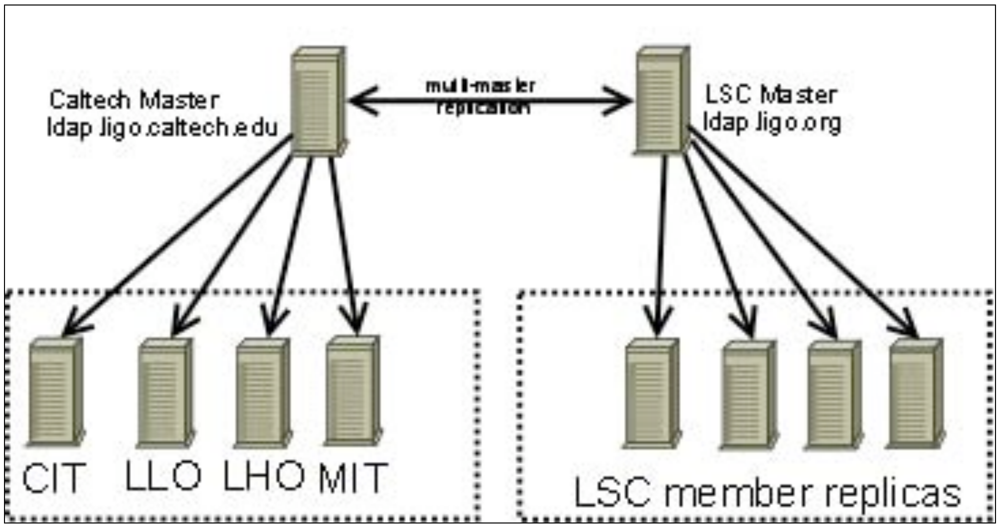
[14] http://www.faqs.org/rfcs/rfc2798.html

The orgPerson, its extension the inetOrgPerson, and the eduPerson seem to be the most common objectClasses used to define people. We anticipate that we will likely continue to use the inetOrgPerson to define people in the roster.

Replication

Replication between LDAP servers varies according to the server implementation. The basic types of replication are multi-master, master-slave, or a combination of the two.

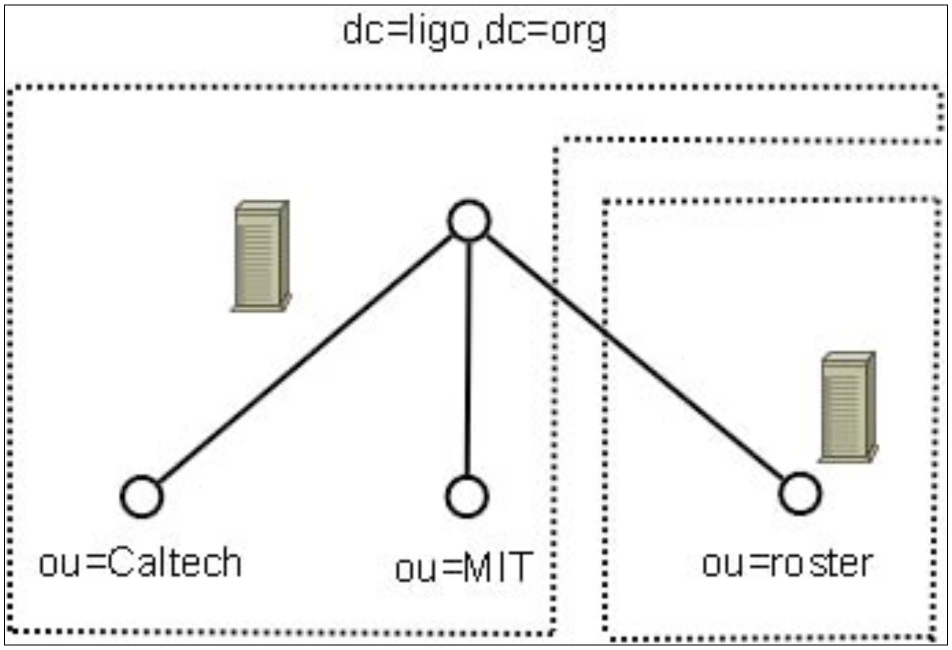
It is thought that we would want to support at least some type of multi-master replication. Having multi-master replication would allow individual groups to modify their copy of the directory and then have those changes replicated to other masters and slaves. Changes would not be limited to one server under control of one or more admins. An illustration of multiple masters each with slaves is below.



There is generally a limit on the number of master servers that can be configured. Both the Sun and RedHat/Fedora servers have a 4 master limit. OpenLDAP does not currently support multi-master replication, though support is anticipated with a future release.

One directory server or Partitioned directory servers?

It is possible to divide portions of the directory tree between servers. Different servers can be responsible for different portions of the directory tree. Below is an example of two servers, each handling a different portion of the dc=ligo,dc=org directory.



We have yet to investigate this type of "federated" configuration and its implications including how it would affect replication between servers.

Performance, scalability

With an organization with <1,000 members, we do not expect scalability or performance will be an issue regardless of the server implementation that is chosen. The iPlanet based directory servers (Sun LDAP, RedHat/Fedora) have been shown to scale to millions of entries and a significant number of queries per second. A number of benchmarks are available on the web. Several years ago, the iPlanet based servers were able to perform better than OpenLDAP, though this may no longer be the case.

Ease of management

The various LDAP implementations may or may not come with their own management utilities for management of the directory. However, LDAP is an open standard, and as such there are both open source and commercial tools for management of the DIT. The standards specify the protocol, therefore anyone can develop tools to interact with the LDAP servers. The list of tools varies from basic command line scripting tools to full blown commercial GUIs. The standard format for importing and exporting LDAP data, the LDIF file, should be supported by nearly any tool available for LDAP management. A list of example tools is in the footnote.[15]

[15] Examples of LDAP management tools are: OpenLDAP.org utilities http://www.openldap.org/; Softerra LDAP Administrator http://www.ldapadministrator.com/; PHPLDAPAdmin http://phpldapadmin.sourceforge.net/; Sun directory management console http://www.sun.com/software/products/directory_srvr_ee/dir_srvr/index.xml

Support availability

Both the Sun and RedHat directory servers offer commercial support contracts. Novell's offerings likewise. There exists an extensive list of vendors who offer commercial support for OpenLDAP. See http://www.openldap.org/support/ for a listing. There is extensive documentation for all versions on the web, so self support is also relatively simple.

4.9 Directory Services

LSC Directory Services was deployed in March 2007. It was designed to support communication and secure information sharing across the LIGO Scientific Collaboration.

The Directory Services back-end includes the definitive database of LSC members, including their institutional affiliations, contact information, certain group roles (e.g., PI, council delegate), collaboration roles (e.g., offices and committee memberships), demographic, and LIGO-effort information required to determine authorship eligibility, group council delegate allocations, and shift responsibilities.

From this information Directory Services generates an always-current directory of LSC members, always-current e-mail lists that reach LSC members and sub-groups (e.g., principal committees, PIs, graduate students, etc.), individualized e-mail forwarding service for LSC members, group MoU Attachment Z reports, etc.

Directory Services includes a secure, web-based interface that regular members use to manage their membership within the collaboration (joining, changing affilitation, changing contact information) and PIs use to manage their group within the collaboration (e.g., effort and demographics reporting, adding and removing members, assigning roles, etc).

The back-end database exposes, via LDAP, contact information to e-mail clients.

Directory Services will play an integral role, with Shibboleth middleware and the other tools discussed earlier, addressing universal LVC-wide authentication and authorization. The Directory Services infrastructure uses a web-based PHP front end to an SQL database, a Kerberos KDC, and an LDAP database fed from the SQL database. Any changes made through the PHP front end are immediately reflected in the SQL database and, within one minute, in the LDAP database. The Shibboleth middleware would be able to take advantage of the Directory Services interface and LDAP database to alter member attributes and provide a basis for authorization privileges throughout the Collaboration.

An example (draft) LVC User Schema follows.

DRAFT USER SCHEMA GOES HERE++++++++++*****************************

At present, the directory includes only members of the LSC. It will need to be expanded to include LIGO Lab personnel and others who need frequent or occasional access to LVC computing resources. The latter category is diverse, including computing maintenance personnel, external computing development collaborators, and our special relationship with Virgo.

Virgo personnel could be included in an integrated LVC database, expanding on the present LSC roster and including Virgo personnel, or Virgo could be included via a federation to a Virgo maintained roster database. In the latter case, the database could either be a replication of the LIGO roster tools with Virgo names, or it could be an expanded schema version of the existing Virgo database (https://pub3.ego-gw.it/itf/Members_DB/Members_List_Public.php) implemented with an LDAP interface. Both possibilities would be federated in the LDAP space as discussed in the earlier LDAP discussion. Decisions on how this would be implemented will be worked out with our Virgo colleagues.

Essential to the effectiveness of the new authentication/authorization plan we propose is that the underlying directory be an accurate and timely reflection of the actual membership of the LIGO-Virgo Community, and the privilege and group attributes of individual members. This will require education of the community about policy, and what the collaboration and laboratory leadership expects of the membership, in this area. Inevitably, keeping the roster accurate and timely will be based on the attention and cooperation of all community members and group leaders.

5 Community Computing Services

In this section we describe the details of authenticating to the various services that directly support the community's activities.

5.1 Apache web server

Background

Apache is web server software used for serving web pages. Approximately 50% of all web pages on the Internet are served by the Apache httpd server[16] with other software packages from Microsoft, Sun, and other vendors making up the other half. Apache is traditionally hosted on UNIX based operating systems, though it will also run under Windows. Apache is also the most common web server software currently in use in the LIGO Lab and the LSC and probably also Virgo.

Apache is easily extensible partly due to its open source code base.

Shibboleth support

From the Shibboleth FAQ[17]:

> *What is required for a campus to set up and to operate Shibboleth?*
>
> *There are different requirements to operate a Shibboleth target or a Shibboleth origin. It is advised that origins operate some manner of SSO service. Shibboleth is currently only available as an Apache module, so targets must operate Apache web servers. No other requirement is particularly unusual and most should be found in common deployments, but for a detailed list of requirements, refer to the Shibboleth Deployment Guide.*

Since Apache is already the "standard" web server software in use within the LIGO world and probably the rest of the LVC, this is not seen as a major problem. For services that will not run on Apache, it may be possible to implement an Apache proxy to take advantage of the SSO functionality of Shibboleth (unverified at the time of this writing…) though this would depend on the application being proxied among other details.

Authentication backends

Apache supports quite a number of authentication backends, too many to list and detail fully in this document. A partial list would include simple text files, LDAP, Kerberos, SQL databases, etc.

Essentially, any Apache module which is capable of providing authentication services can be used as an authentication backend for Shibboleth[18].

[16] http://news.netcraft.com/archives/web_server_survey.html

[17] http://shibboleth.internet2.edu/shib-faq.html#11

[18] https://spaces.internet2.edu/display/SHIB/IdPUserAuthnConfig

In Shibboleth 1.3, the interaction between the IdP and an Apache-based local authentication system is implemented by providing the SSO handler with the identity of the browser user through REMOTE_USER . Any authentication system that is capable of protecting a block of webspace using httpd.conf for the SSO handler and populating the REMOTE_USER header variable is compatible with Shibboleth.

5.2 Wikis

Background

Wikis have become a valuable collaboration tool allowing collaborative edits of web pages. Wikis have sprung up in various places within the LVC, usually with the local sysadmin or user determining which wiki software to use. Most wiki software implements their own authentication mechanisms, some support pluggable mechanisms, many reinvent the wheel and only support their own mechanisms.

It would be desirable for the LVC to standardize primarily on authentication and authorization for wikis in use, so that it eases the burden on users for authentication. It is also desirable that authorization be performed in a standardized manner, likely based on "group" membership.

Many wiki implementations support at a minimum apache authentication, with some supporting full shibboleth[19] authorization schemes. Included later is a description of several wiki implementations and the level to which they support shibboleth Authentication/Authorization. As we presently understand it, if the wiki application can understand apache authentication, then by extension, it can support shibboleth SSO. There may be exceptions to this statement.

Some wikis by default do not perform any authentication "out of the box", or they do not place restrictions on who can and cannot create user accounts. It has been necessary to enable account restrictions on wikis in the LSC due to defacements including porn and spam postings. The method of placing account restrictions on wikis depends on the particular software in question, and these restrictions have been met with varying levels of success.

5.2.1 Individual Wiki implementations

Confluence

Confluence is a commercial product, which supports "full shibboleth integration"[20] via an open source plug-in.

A list of features from the vendor's website is listed below.

[19] "Full shibboleth integration" is not always clearly defined. We presently understand it to mean that the application, wikis in this case, can accept more variables than just REMOTE_USER from apache at the "Service Provider" location. Things like "Shib-InetOrgPerson-mail" and "Shib-InetOrgPerson-displayName" can be passed on to the application from the shib-IdP to define the user's full name and email address.

[20] http://confluence.atlassian.com/display/CONFEXT/How+to+Shibbolize+Confluence

http://www.atlassian.com/software/confluence/

- Enterprise security
- Simple installation and management
- Attractive, user-friendly WYSIWYG interface
- Powerful tools for structuring and searching your wiki
- Professional features such as PDF export and automated refactoring
- An open API for extension and integration
- Atlassian's Legendary Service
- and much more...

Confluence is a feature packed wiki and looks attractive for LVC use. However it is a commercial product, though the price seems reasonable enough to consider. Pricing information is available on their website. Annual .edu licensing fees for an unlimited user license is $2000; 500 users for $1000.

5.2.2 Mediawiki

Mediawiki is the wiki software used at Wikipedia.com. It is an open source GPL licensed wiki. It is written in PHP, has a MySQL backend, and is used commonly on the web. By default, mediawiki has its own authentication mechanism, which is based on users stored in a MySQL database. There are third party plug-ins which will allow both shibboleth and apache based authentication for Mediawiki.

See:

http://www.serdarbulut.com/MediaWiki-and-Apache-Authentication-Integration

http://www.mediawiki.org/wiki/Extension:Shibboleth_Authentication_Plus

With the apache plug-in, and presumably with the Shibboleth plug-in, accounts are created in the MySQL database automatically, if they do not exist after external authentication is completed.

5.2.3 Moinmoin wiki

MoinMoin is another open source wiki commonly used on the web. By default, the wiki does not require user authentication to edit pages, nor does it restrict account creation. Hanford has worked with MoinMoin and restricted account creation and restricted edits of wiki pages with varying degrees of success. It has proven problematic.

MoinMoin claims to support apache authentication, and therefore Shibboleth SSO, though we have not investigated this in detail.

From the MoinMoin documentation[21]

[21] http://moinmoin.wikiwikiweb.de/HelpOnAuthentication

How Authentication works with MoinMoin

MoinMoin historically has used some cookie-based authentication: you log in via the form on page UserPreferences, moin sets a cookie and from then on this cookie is used for authenticating you - until you log off and the cookie gets deleted (or until the cookie expires).

For running moin in corporate environments this is often no option as access restrictions have to be enforced reliably. Starting with 1.3 moin could also use HTTP basic auth based authentication, when being run with some web servers (like Apache) supporting it.

Starting with 1.5 moin now has freely configurable and kind of modular authentication. You use the auth configuration value to set up a list of authentication methods that are processed in exactly that order.

When an external user database is used you do not want to recreate all users in moin. For this case the configuration option user_autocreate was added to moin 1.5. If you set it to True a new user profile will be created automatically when a new user has passed authentication (and the auth method supports auto creation).

5.2.4 Wikkawiki

WikkaWiki is in use on CDS at LLO.

5.3 Other network based tools

We discuss here other important collaboration network based tools that do not work through a web interface.

5.3.1 Version Control Systems

The two primary version control systems being used within the LSC currently are CVS and subversion (SVN).

5.3.2 CVS

We are not aware of any direct use of cvs over http, and thus shibboleth authentication and authorization through the web is not possible. The alternatives are:

1. Via Remote Shell Access - CVS can interact with remote repositories over a remote shell connection. This access method is selected by the user as part of the repository URI string they enter on the command line. For example:

   ```
   $ cvs -d :ext:username@remote.host:/repository/path checkout package
   ```

   The ":ext:" requests that cvs use an external rsh client to communicate with the remote host. By default the cvs client runs "rsh", but a different program (e.g., gsissh) can be selected by setting the CVS_RSH environment variable. Using this access mechanism, cvs authentication/authorization can in principle piggy-back on x509 certificate-based remote shell access, Kerberos remote shell access, or any other "single-sign-on" system for remote shell access, as discussed above.

2. PAM - An alternative to communicating with a repository over a remote shell connection is to speak directly to a CVS server daemon. The cvs server daemon can use PAM for authentication. Unless told not to, the server daemon will fallback to the system's user look-up routines if it cannot authenticate a user using its own password files, and on Linux systems that means falling back to PAM. Typically, the file /etc/pam.d/cvs will contain the PAM rules for the cvs server. Via PAM, it should be straight-forward to configure a cvs server to authenticate via Kerberos.

5.3.3 SVN

Subversion-based revision control can be performed using several underlying protocols including http. On the server side http-based access to the repository can be provided by Apache using the "mod_dav_svn" module. This means that an svn repository can use shibboleth for authentication and authorization, even trivially if the apache server in question is already configured to protect other pages with shibboleth. However, the command-line svn client tool does not support the cookies, redirects, and http POST mechanisms needed to perform shibboleth-based authentication with the server (although it does support https).

The paper http://doi.ieeecomputersociety.org/10.1109/ITNG.2007.201 presents a report of the implementation of a web interface providing read/write access to a subversion repository for collaborative document preparation using shibboleth for authentication and authorization. It's not clear from that document, but it appears that this is meant for revision control of single files where the user accesses a web page with a browser and can commit a new version or check out the latest version and so on, via cgi scripts. That setup is not suitable for revision control of entire source trees.

In summary, full read/write access to a subversion repository using shibboleth-based authentication and authorization is (almost certainly) a solved-problem at the server side, but altogether appears to be impossible without significant modifications to the subversion client code. Alternatives are:

1. Via Remote Shell Access - Like cvs, the svn client program can access remote repositories through a remote shell and so authentication/authorization can piggy-back on x509 certificate-based shell access or any other single-sign-on system for remote shells. Again, the user selects shell-based remote access as part of the URI string entered on the command line. For example:

   ```
   $ svn --username username checkout svn+ssh://remote.host/path/to/repository/package
   ```

   By default, svn executes "ssh", but the SVN_SSH environment variable can be used to select a different program (e.g. gsissh). When the svn client invokes ssh, on the ssh command line it sets the remote command to execute (i.e. svnserve), which the sshd server executes instead of a login shell. Therefore, it is not necessary for the user to have a home directory on the remote machine, and their default login shell can be set to /bin/false (which only disables accidental shell access, it must be understood that users do have shell access to the machine hosting the repository). When accessing an svn repository over an ssh tunnel, it is the username attached to the account on the server to whom the repository revision history will attribute modifications, so the user should have a unique account and it's name needs to be that user's normal ID. For example, it is not practical to use a gridmap file to map all remote users to a single local SVN account on the server, nor is it appropriate to set the usernames to random machine-generated sequences of letters and numbers.

5.3.4 GIT

Normally there is no central repository when using GIT for revision control. Each working copy is a full repository with a full local revision history, and users only ever pull work into their own repository from other repositories. To collaborate, each user creates their own repository and exposes it to others for read-only access, and users then pull modifications from each other when they desire them. Therefore, the concepts of authorization and authentication are a little different in the context of git. In the normal mode of operation, nobody has write permission to a repository except the user who created it, and so there is no question of how to selectively grant write permission to remote users.

At the same time, since every user is running their own repository and will want to expose it for read access so that others can retrieve their contributions, and since people might be collaborating on sensitive material like a paper draft that should not yet be made public, there is a desire to selectively grant remote read permission to a repository. This has to be solved not just once for a single central server, but it has to be solved over again for every person in the collaboration who wishes to expose their own working copy.

The GIT documentation specifically discourages it, but it is possible to use GIT in a CVS- or SVN-like mode where there is a single shared common repository into which all users can push revisions. This is done by creating a repository on a machine where all collaborators have login access, and giving them all read/write permission to the files in the repository.

GIT repositories can be exposed for remote read-only access with a web server, and so like subversion it is in principle possible to protect these repositories with Shibboleth. However, we have been unable to find any indication that the GIT client supports the http features required to enable shibboleth-based authentication (cookies, redirects, POST, although it does support https) and so like Subversion it appears that Shibboleth cannot be used to protect GIT repositories without modifications to the client source code.

The alternatives to Shibboleth for GIT:

1. Remote Shell Access - Like CVS and SVN, GIT can access remote repositories over ssh. The documentation does not state this, but it gives the impression that this is in fact the only way to "push" local modifications into a remote repository, so presumably this would be the normal access method for GIT users in the LVC. By default GIT executes "ssh", but the GIT_SSH environment variable can be used to select a different program (e.g., gsissh). In this way, access to centralized GIT repositories can piggy-back on whatever single-sign-on technology is adopted for remote shell access.

5.3.5 Command-line HTTP clients

For scripting purposes, it is sometimes convenient to use the command-line HTTP clients, wget and curl.

wget

wget supports https, cookies, and can fill out forms using the http POST mechanism. However, wget will only send the http POST data to the first web page it visits in a chain of redirects. Therefore, if it is possible to perform a Shibboleth login by connecting directly to the identity provider's username/password page instead of being redirected there from the page one wishes to view, then wget can be used to access data behind shibboleth-protected pages. Example (adapted from wget man page):

```
# Login by connecting directly to the identity provider's login page,

# sending POST data, and storing the cookie that gets sent back

wget --save-cookies cookies.txt --post-data "user=foo&password=bar" \

https://id.server.edu/auth.php

# Now access the protected content, using the cookie sent from the identity provider

wget --load-cookies cookies.txt https://data.server.edu/protected/data.php
```

The cookies.txt file can be re-used, accessing multiple protected pages onmultiple servers until the cookie expires.

curl

Like wget, curl does not handle redirects well. We have not investigated at this point whether a mechanism similar to the wget method outlined above works, but see no reason in principle why it should not.

6 Appendix I – LSC Directorate Charge

From: Lazzarini Albert <lazz@ligo.caltech.edu>

Date: November 29, 2007 11:12:56 AM PST

To: T Nash <nash@ligo.caltech.edu>

Cc: LSCComputingCommittee <lsccompcomm@ligo.caltech.edu>

Subject: [Lsccompcomm] Directorate response to the Authentication and Authorization Subcommittee's recommendations

Tom,

Thank you for the cogent overview of the issues confronting the LSC and the Lab in the area of access authentication and authorization ("A/A") to data and IT resources. You presented a number of compelling arguments for a broad and urgent need in this area and why it is advisable to establish close linkage to the ongoing Directory Services Project in order to provide infrastructure to support A/A. We agree that this is an issue that needs to be solved sooner rather than later.

Speaking from direct experience, we also realize that the solution is likely to be more complex than what is anticipated today. For this reason, we endorse the idea of mounting an intense near-term effort to put together rapidly a prototype demonstrator with all the needed features. We expect that this effort will require a concerted effort from a number of already-very-busy individuals; yet, the need is real. We are prepared to provide direct impetus to make this happen: LIGO Lab will work to assign the individuals who are identified as crucial to help develop the prototype; the Directorate will _strongly encourage_ others who are needed from outside the lab to take part. The immediate next step needed is for the Authentication and Authorization Subcommittee to put together a list of names for a rapid-prototyping task force, to develop a timeline for the prototype, a definition of the minimum level of success, and to identify the venue.

We await these before either the Lab or the Directorate will take further action.

Looking to the larger challenge beyond building a demonstration prototype, it is equally apparent to us that this effort is a campaign: i.e., it is a full-fledged project. In order to maximize the chances for its rapid deployment, acceptance, and success, we believe very strongly that the effort needs to be organized as a project: it should not be "run by committee." By this, we mean the following. Programmatic oversight and guidance should remain the purview of the Computing Committee, since much of the strategic insight resides there. HOWEVER, there needs to be a task manager -- an “A/A Czar” -- whose job it will be to make this happen and who is accountable to the Directorate for the duration of the project. It is our intent to provide the full support of the Directorate in order to empower this individual to enable him or her to accomplish the task before us. As you work now to rapidly prototype a demonstrable solution, we ask the Computing Committee to put forward an A/A Czar candidate.

The first job for the person who accepts this role will be to put together the master plan for how the "campaign will be won." That plan should identify those sites which will be involved, the names of individuals who will be dedicated to the A/A project by these sites, their % FTE commitments, etc. We also want to see a deployment plan: how will the roll-out happen? Who are the "guinea-pigs"? What is the "outreach" plan to gain overall LSC acceptance? How will it be maintained? What are the (fiscal and human) costs -- how will we achieve these goals within the reality of all other constraints on manpower, time, etc.? Finally, how will we deal with our EU (both intra- and extra-LSC) colleagues?

Thanks,

Albert, Jay & Dave

LIGO Laboratory

California Institute of Technology M/S 18-34

1200 E. California Blvd.

Pasadena, CA 91125

626-395-8444 (Office)

626-304-9834 (Facsimile)

7 Appendix II – Current Access Controls

We currently implement a wide variety of access controls for the services deployed. A partial list is:

- **Web services**
  - static web pages use individual login/password, individual login/shared password, shared login/password, and X.509 certificate authentication via Apache htaccess modules. There have been cases of restricted access material being posted without access controls on URLs that administrators believe obscure enough that they will only be accessed by users who have been informed of the URL – i.e. security through obfuscation. Where individualized user passwords exist, they are generally not coordinated between services, and Apache does not have a built-in mechanism for users to change their passwords, resulting in large numbers of passwords for users to remember. Authentication via X.509 certificates requires users to import these certificates into their web browsers, which has a whole list of associated problems (certificates must be put into the proper form, trusted root certificates must also be imported, must be repeated annually when certificates expire, etc).
  - enotebooks, inotebooks, elogs have shares or individual passwords. Users cannot usually change their password.
  - wikis use individual or shared passwords. Users can usually change their password. Also, wikis which host private information often must be in Apache htaccess controlled spaces, or they will be publicly viewable. This means that users often must enter two different credentials in order to update a wiki.
  - mailing list management pages (archives, etc) use individual passwords, which are user changeable.
  - the document control center
  - the LIGO roster uses individual passwords, which are user changeable.
- **General Computing (GC) services**
  - console login uses either user-changeable standard PAM username/password authentication against shadow password files or against LDAP. Coordination of account information is achieved between systems with NIS+ except at LLO where LDAP is now implemented.
  - ssh between systems uses standard shadow password entries and are thus coordinated with console logins.
  - email (imap, pop, smtp) service generally authenticate via POP/IMAP to NIS+, except at LLO where a Sun enterprise product is in use.
- **LIGO Data Grid (LDG)**
  - all grid authentication is handled via X.509 certificates. This provides secure and transparent single sign-on access within this domain. The challenge here has been to have user access reflect LVC membership – the process for new users to get a certificate can be slow and difficult for some users, and there is currently no mechanism for removing user access when someone leaves the LVC, in general.
- **Other Services**
  - version control systems access is currently controlled by an intrinsic password mechanism (passwords are not user changeable) or in some few cases have been

  - delegated to ssh. In the latter case, grid-enabled ssh allows for transparent access for those with grid credentials.
  - mailing lists membership is managed by user enrollment, either through self-subscriptions which are approved by list administrators, or by mass enrollment by the administrators themselves. Currently, the diligence of list administrators is the sole mechanism to insure that membership is current and correct.
- **Control and Diagnostic Systems (CDS) services**
  - currently, many access controls listed for other services is employed in CDS. As well, there are specialized systems, which implement proprietary access controls. There are some services for which physical access to the host system is the only access control. In some cases, shared account are used. The highly specialized and heterogeneous nature of CDS systems makes coordination of access controls in this domain difficult except at the gateway box isolating CDS from offsite access.

8 Appendix III - Native LDAP NIS+ replacement for Solaris

```
dn: cn=Directory Administrators, ou=ligo
objectClass: top
objectClass: groupofuniquenames
cn: Directory Administrators

dn: ou=people,ou=ligo
ou: people
objectClass: top
objectClass: organizationalUnit

dn: ou=group,ou=ligo
ou: group
objectClass: top
objectClass: organizationalUnit

dn: ou=rpc,ou=ligo
ou: rpc
objectClass: top
objectClass: organizationalUnit

dn: ou=protocols,ou=ligo
ou: protocols
objectClass: top
objectClass: organizationalUnit

dn: ou=networks,ou=ligo
ou: networks
objectClass: top
objectClass: organizationalUnit

dn: ou=netgroup,ou=ligo
```

```
ou: netgroup
objectClass: top
objectClass: organizationalUnit

dn: ou=aliases,ou=ligo
ou: aliases
objectClass: top
objectClass: organizationalUnit

dn: ou=hosts,ou=ligo
ou: hosts
objectClass: top
objectClass: organizationalUnit

dn: ou=services,ou=ligo
ou: services
objectClass: top
objectClass: organizationalUnit

dn: ou=ethers,ou=ligo
ou: ethers
objectClass: top
objectClass: organizationalUnit

dn: ou=profile,ou=ligo
ou: profile
objectClass: top
objectClass: organizationalUnit

dn: ou=printers,ou=ligo
ou: printers
objectClass: top
objectClass: organizationalUnit
```

```
dn: ou=SolarisAuthAttr,ou=ligo
ou: SolarisAuthAttr
objectClass: top
objectClass: organizationalUnit

dn: ou=SolarisProfAttr,ou=ligo
ou: SolarisProfAttr
objectClass: top
objectClass: organizationalUnit

dn: ou=Timezone,ou=ligo
ou: Timezone
objectClass: top
objectClass: organizationalUnit

dn: ou=ipTnet,ou=ligo
ou: ipTnet
objectClass: top
objectClass: organizationalUnit

dn: automountMapName=auto_home,ou=ligo
automountMapName: auto_home
objectClass: top
objectClass: automountMap

dn: automountMapName=auto_direct,ou=ligo
automountMapName: auto_direct
objectClass: top
objectClass: automountMap

dn: automountMapName=auto_master,ou=ligo
automountMapName: auto_master
```

```
objectClass: top
objectClass: automountMap

dn: automountMapName=auto_shared,ou=ligo
automountMapName: auto_shared
objectClass: top
objectClass: automountMap
```