The GridAuth Credential Management System *

Timothy Warnock, Wei Deng, Lawrence Miller
San Diego Supercomputer Center
9500 Gilman Dr MC 0505
La Jolla, CA, United States, 92093
{twarnock|weideng|ljmiller}@sdsc.edu

Adam Lathers
University of California San Diego
9500 Gilman Dr
La Jolla, CA, United States, 92093
alathers@ncmir.ucsd.edu

ABSTRACT

Grid savvy credential management is a vital component in the process of developing user accessible grid resources. In order to facilitate secure single sign-on across distributed systems a trusted credential repository can be leveraged by multiple security domains. GridAuth manages user credentials in a secure system allowing for seamless integration with existing distributed resources. This paper describes the GridAuth architecture, design, implementation and deployment. GridAuth is currently deployed in NEES[14] to manage all authentication, authorization and auditing.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Authentication, Access Control; H.3.5 [**Online Information Services**]: Web-based services; H.3.4 [**Systems and Software**]: Distributed Systems

Keywords

Grid, Authentication, Authorization, Auditing, Security

1. INTRODUCTION AND RELATED WORK

GridAuth is a user credential management system for distributed data and computational resources. Using a plugin-based architecture, GridAuth is configurable and extensible; making it adaptable to any system requiring credential management, advanced authorization and secure authentication.

As distributed grid environments such as NEES[14] and GEON[8] continue to evolve, centrally controlled credential management systems are being developed to abstract traditional GSI[6] services such as MyProxy[16] and CAS[18]. Web portal solutions, such as GAMA[7] and PURSe[19] demonstrate a growing need in virtual organizations to remove the complexities of X.509[11] certificate management and provide simple means for users to authenticate.

GridAuth provides a deployed authentication, authorization and auditing system with an extensible architecture useful in web portal environments as well as any distributed computational or data environment such as SRB[22] and Globus[5]. Furthermore, GridAuth leverages existing standards such as GSI[6] as well as working to evolve future standards with related technologies such as SAML[20].

2. PROJECT MOTIVATION AND OVERVIEW

2.1 User Problem Statement

Grid security and credential management is difficult to the point of unusable for both users and developers. Few tools or APIs exist for developers to quickly add authentication and authorization of grid credentials into an application. Users who expect a simple password-based scheme must instead manage their own certificate and key and understand the mechanisms to manually delegate temporary proxy certificates[23]. GridAuth provides a turnkey solution for building an effective and user-friendly GSI[6] system.

2.2 Use Cases

The account and user credential system provided by GridAuth supports a variety of use cases.

- **Authentication** One of the most challenging security problems facing distributed grid environments is authentication. Authentication is the process where a user proves his or her identity; that they are who they say they are. GridAuth relies on GSI[6] authentication leveraging public-key cryptography and mutual authentication.
  GridAuth extends the standard GSI[6] model by introducing user-friendly sign-on mechanisms where authentication is abstracted from the user point-of-view.

*This work was supported primarily by the National Science Foundation under Award Numbers CMS-0117853, CMS-0086611, CMS-0086612.
Permission to make digital/hard copy of part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date of appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
2005 ACM 1073-0516/01/0300-0034 $5.00

Using GridAuth, users simply login with a username and password, all GSI[6] credentials are centrally managed behind the scenes without the users' direct involvement.

- **Authorization** Authorization is the process of granting or denying access to a resource. This is a critical component in any distributed environment where resources often cross various security domains. GridAuth provides role-based authorization with unlimited roles as needed by the given user community. Furthermore, a given account can be affiliated with unlimited roles.

- **Auditing** One of the core principles to any security infrastructure is the process of auditing; securely and thoroughly recording authentication and authorization requests. In distributed grid environments this is a challenging problem due to the many resources spread across various security domains. Most GSI[6] models do not provide central auditing, which poses significant risk to all resources involved in a grid. By maintaining a centrally controlled distributed security infrastructure, GridAuth provides a central facility for logging all authentication attempts and provides a method to allow remote applications to send messages to a logger service.

- **Single-Sign-On** In addition to basic authentication, an authenticated user can maintain their session across a distributed grid-based environment. GridAuth utilizes centrally controlled session handling to allow single sign-on. In practice, this means that a user can login through any possible entry point in a distributed grid and access all resources that they are authorized to access.

- **Delegation** To allow a user access to a resource that they are authorized to use, but exists in an external security domain, a process of delegation must exist to allow the authentication and authorization to be recognized without compromising any centrally managed security infrastructure. GridAuth extends the GSI[6]-based X.509 proxy certificates[23] with the addition of short-lived session-ids that map one-to-one with each authentication instance and each proxy certificate[23].
  Upon successful login to a GridAuth system, a session id is generated along with a valid X.509 proxy certificate[23]. A GridAuth session is a unique hash sequence that is transmitted to remote security domains over TLS and is only valid for the duration of the users' session. Once the session id has been validated on a remote system through the centrally controlled GridAuth server, then the username, X.509 proxy certificate[23], access roles, and other account credentials are available to the remote application.

- **Account Synchronization** Distributed grid environments, by design, involve shared resources spread amongst various security domains. In many cases, legacy applications are shared with newer GSI[6]-based applications and have no built-in method to access the basic authentication and authorization provided by GSI[6]. GridAuth helps to alleviate this problem by providing an extensible plugin framework to extend the basic GridAuth functionality by writing custom plugins.

- **Account Management** In a real-world deployment an effective and robust account management system must be provided to allow administrators to manage accounts. GridAuth provides a web-based account request system where accounts can be requested from a web-portal interface. An account management portal is provided where each request can be confirmed or denied. All details of the underlying security infrastructure, such as an X.509 certificate and key[11] are abstracted from this interface.

2.3 Requirements

The design, implementation, and deployment of GridAuth meet the following requirements.

- **Client System** No Globus[5], GSI[6], or other security software needs to be installed on the end-user system.

- **Server System** A traditional LAMP (Linux, Apache, MySQL, and PHP/Perl) platform is recommended although not required. The core GridAuth service is provided through Perl modules including Perl DBI/DBD for all database access. Globus[5] and OpenSSL[17] are required for full GSI[6] support.

- **Simple username/password Login** All end-user authentication must be handled through a simple username and password login. Furthermore, no middleware or GSI[6] handling should be required for logging into the system.

- **Single Sign-On** Any resource that can be accessed by a single username and password login must also be accessible to users who have already entered their username and password. There must be no dependence on where an account was authenticated. Users should be able to authenticate anywhere on the grid and access any resource they are authorized to access for the duration of their session.

- **Forgot Passwords** Passwords can be reset in an automated fashion, for example, emailing the user for account verification.

- **Role-Based Access** Access to any resource can be controlled, by approving or denying specified roles rather than on an account by account basis.

- **Application Neutral** No specific application platform is required for integration with distributed grid resources. For example, web portal interfaces may be used along side client applications.

- **No Local User Accounts** GridAuth users exist in the abstracted domain of a distributed grid, that is, the collection of all distributed resources in a virtual organization. As a result, local accounts are not required although may still be used to map access roles.

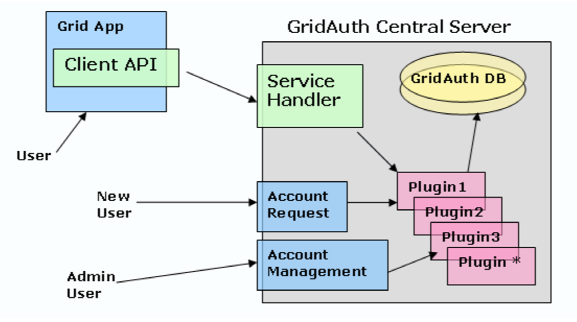

Figure 1: GridAuth Architecture

- **Minimize grid-mapfile Maintenance** Traditional GSI[6] solutions require maintaining grid-mapfiles across a diverse set of resources for every account in the grid. This process is not scalable and presents an inordinate burden on system administrators. There may be role accounts for specific resources as required by a specific application.
- **No User Managed Credentials** GridAuth provides central management for distributed resources and users. In the regular GSI model, the users maintain grid credentials and generate the corresponding proxy credentials on their local systems, which are not necessarily trusted or well-protected. In the GridAuth framework, credentials are managed on a trusted and secure system with limited access, where security protection is easier to enforce and monitor. With centralized management of user credentials, it also has the benefit of lifting the burden on the regular users to maintain acceptable security for their credential files.

3. ARCHITECTURE

GridAuth provides client APIs used by external applications or grid nodes; these APIs communicate with a central service handler that manages the plugin stack. Plugins are completely customizable and configurable to facilitate migration issues with existing grids as well as advanced grid specific resource synchronization. Account management is provided through both an administrative web interface as well as a command line interface. The framework is shown as in Figure 1.

3.1 Client API

The client API provides high-level functions to application developers for authentication and authorization. The underlying implementation details are all handled on the server side making the client API extremely lightweight, secure and portable.

For example, if a specific GridAuth instance was utilizing MyProxy[16] and LDAP[24], the application developer does not need to interface with either of those systems directly. Instead the API provides high level functions, such as login and logout. The service handler will interface with specific plugins for MyProxy[16] and LDAP[24], returning the fully transparent results to the client API.

This abstraction is helpful to minimize the load on developers; it allows for easy migration of existing services, and helps to enforce consistency in the requirements and application of security policies. Furthermore, this allows forward and backwards compatibility with specific authentication and authorization infrastructure.

3.2 Service Handler

The service handler accepts requests from any client API and will pass off the request to the plugin stack. The service handler maintains a customized list of plugins defined for the specific grid instance. All network transactions occur over a secure SSL[3] port, sending XML[1] responses via HTTP[4].

3.3 Plugin Stack

A GridAuth instance includes a plugin stack made of existing and custom built plugins, each one implementing required interface functions (such as login and logout). For example, when a client API sends a login request, the service handler will call login methods, in an ordered fashion, on each one of the plugins in the stack. Upon successful login (all plugin calls return true) the response is sent back to the client API via the service handler. The exact plugin design is defined in section 5.1.

3.4 GridAuth Database

The GridAuth database is provided to allow for persistent storage of session information in addition to permanent storage of user account information. The plugin design (section 5.1) provides all plugins with access to this database. Not all plugins will utilize the database, but is provided for convenience and performance.

3.5 Account Request

New users can freely request accounts via a simple web interface. The web interface is defined by the plugin stack. In this manner, plugins can be defined to gather information specific to a grid instance or organization. For example, a plugin could be provided to request for height, weight and age if this was something worth capturing (e.g. clinical patient grid). All information must be stored pending account approval or denial.

3.6 Account Management

Account management is provided via command line tools and additionally as a web interface. The administrative user can list all active grid accounts and modify or delete as needed. Account requests can be viewed, edited, activated or deleted. Any account actions will trigger the corresponding method in the plugin stack. For example, upon account creation, every plugin in the stack will call its account creation method.

4. DESIGN ISSUES

GridAuth provides multiple client API's that use standard HTTP[4] over SSL[3] to communicate with a central server. A GridAuth central server handles all authentication and authorization requests for all grid sessions. Additionally, the central server is responsible for all auditing requests.

To facilitate a custom grid environment, plugins can be used to offer grid administrators total flexibility. When a GridAuth instance is installed, specific plugins are registered

through a simple configuration process. The service handler will call every registered plugin upon each transaction. The service handler will call these plugins in the order they are registered.

All plugins have access to an SQL compliant relational database (default MySQL) to store permanent and temporary information. This includes in-memory information about active sessions as well as permanently stored user credentials.

4.1 System Requirements

Any system capable of running Apache, Perl, and MySQL will be able to run GridAuth. However, individual plugins may have more specific requirements. For example, a certificate authority plugin may require OpenSSL and Globus[5]. It is the responsibility of the plugin author to strictly define the system requirements for the plugin.

4.2 Integration Issues

The core GridAuth module has no impact on existing software. However, individual plugins may have direct impact on other software. For example, a tmpfs file system mount may be required for a proxy[23] store plugin. It is the responsibility of the plugin author to identify integration issues and dependencies.

4.3 Security Issues

All remote access is strictly confined to the service handler; the service handler inherits all security issues from the web server. It is strongly recommended that the service handler run over SSL[3] and never over an unencrypted HTTP[4] port.

Grid sessions are uniquely identified with a session key. A session key is a unique SHA1 hash value which is passed from one grid application to another over a secure channel. While intercepting a session key is dangerous, this is more secure than the standard grid convention of delegating proxy certificates[23]. An X.509 proxy certificate[23] is valid for a fixed duration, whereas a session can be terminated as soon as a user logs out.

Currently, a random-seed SHA1 hash value is used to uniquely identify all active sessions. This presents the risk of a brute-force attack to determine active sessions. This risk is minimal since session duration is likely to be far shorter than the computation time required for a brute force SHA1 attack[21], which is on the order of 2**69 hash operations. Stronger hash schemes, such as SHA256 further mitigate this risk.

4.4 User Impact

GridAuth has no direct user interface; however, GridAuth is designed specifically with users in mind. GridAuth allows for a true single-sign-on environment where a user will never need to understand or directly manage grid credentials. For example, rather than explicitly delegating proxy certificates[23], the user simply moves from one application to another, the proxy[23] delegation is performed behind the scenes.

4.5 Risks

GridAuth centrally manages all authentication and authorization services, this presents a logical single point of failure. To mitigate this risk, appropriate measures must be taken to provide fault tolerance on the central server running GridAuth in addition to the GridAuth instance itself.

5. IMPLEMENTATION

5.1 Plugin Design

The plugin design includes the following interface methods; this allows customized functionality without impacting existing code.

```
-useradd(username, userinfo)
-usermod(username, userinfo)
-userdel(username)
-login(session)
-login(username, password)
-logout(session)
-groupadd(group)
-groupmod(group, add|delete, username)
-groupdel(group)
-install()
```

5.2 Command-line Tools

As an alternative to using a web administration page, account management can be performed through the command line. Command line usage is as follows:

```
./gridauth <command> <options>

Valid commands and options include
 useradd <username> {<key>=<value>}
 usermod <username> {<key>=<value>}
 userdel <username>
 groupadd <group>
 groupmod <group> add|delete <username>
 groupdel <group>
 status
 list
```

5.3 Service Handler

The service handler interfaces between the client API and the core GridAuth plugin stack. The service handler supports methods for login, logout and logger (to log arbitrary messages). Standard HTTP[4] POST values can be sent to the service handler. The service handler responds with a simple XML[1] response that is generated by the plugin stack. The API parses the XML[1] response and makes the data available through access methods.

The service handler response DTD:

```
<!DOCTYPE GRIDAUTH [

<!ELEMENT GridAuth (key*)>
<!ELEMENT key (#PCDATA)>

<!ATTLIST GridAuth Version CDATA #REQUIRED>
```

```
<!ATTLIST key name CDATA #REQUIRED>

]>
```

An example service handler response XML[1]:

```
<GridAuth version="1.0">
   <key name="username">jdoe</key>
   <key name="email">jdoe@email.com</key>
   <key name="first_name">John</key>
   <key name="last_name">Doe</key>
   <key name="phone">858.555.5473</key>
   <key name="address">0505</key>
   <key name="comments">no comment</key>
   <key name="groups">nees neesit</key>
   <key name="session">
      b14ce9ec8f8a395130a56e603f025042
   </key>
</GridAuth>
```

5.4 Client API

Currently, client API's have been implemented for Java, Perl, and PHP. Client API specification is detailed in "GridAuth API Specification", available here: http://gridauth.com/?loc=doc

6. PRODUCTION DEPLOYMENT

6.1 NEES Authentication and Authorization

The George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES)[14] integrates 15 national state-of-the-art earthquake engineering experimental facilities with central data and computational services through a unique IT infrastructure, called NEESgrid[12], to link earthquake researchers across the United States with leading-edge computing resources and research equipment, allowing collaborative teams (including remote participants) to plan, perform and publish their experiments.

Early phases of the NEESgrid[12] implementation used a pure GSI[6] model to authenticate users and services, and manage users' credentials and proxy certificates locally. This practice is hard to be understood and executed by regular earthquake researchers, as they are usually not computer savvy users and sometimes don't even know how to use the command line interface of a Unix system and how to manage their credential files. It also requires a lot of maintenance from the system administrators of the resources to manually update the mappings between grid identities and local accounts.

With the integration of GridAuth, the users only need to remember the username and password to authenticate to the GridAuth web portal, and have the central GridAuth server to manage all of their grid credential files. The login web interface on all resources use GridAuth client API so that the users can choose to login at any resource with their GridAuth credentials and get assigned session ID, which can be later used for authenticating at the other GridAuth supported resources. Through the GridAuth client API, the login events at the resources are also reported back to the GridAuth server as audit trails.

6.2 NCMIR Single Sign-On

NCMIR[13] is leveraging the GridAuth package to allow for a smooth integration between multiple in-house and third party software suites. By utilizing this technology, NCMIR[13] is able to provide a smooth interchange between otherwise disjoint software sets. Initial drafts include integration between existing lab resources, the NCMIR[13] Intranet, CCDB[2], and the Neurosys digital lab notebook system[15]. By abstracting the authentication and authorization process away from the user, it is possible to remove the burden of manually joining the functions of all these tools, and instead, allow users to smoothly control data and grid accessible resources like the CCDB[2].

7. CONCLUSION AND FUTURE WORK

The GridAuth credential management system presented in this paper is a flexible and extensible solution for authentication, authorization and auditing across distributed resources. GridAuth is adaptable to several platforms and network-accessible resources including web portals as well as distributed computation-intensive and data-intensive applications. With successful deployment in NEES[14] and integration with NEESgrid[12] software, GridAuth has proven to simplify account and credential management while providing simple login mechanisms and single sign-on across distributed resources.

Future work includes improved security by integrating client-side challenge-response login functions to avoid transmitting sensitive information. Additionally, one-time-password systems may be integrated as well as single-use session identifiers to remove the risk of stolen or compromised session ids. GridAuth is freely available to the public through Sourceforge[10] and from the GridAuth website[9].

8. REFERENCES

[1] J. Boyer. RFC 3076: Canonical XML Version 1.0. IETF RFC Publication, March 2001. http://www.ietf.org/rfc/rfc3076.txt.

[2] CCDB. CCDB: Cell Centered Database. http://ccdb.ucsd.edu/.

[3] N. C. Corporation. The ssl protocol version 3.0. http://www.netscape.com.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol – HTTP/1.1. IETF RFC Publication, January 1997. http://www.ietf.org/rfc/rfc2068.txt.

[5] I. Foster and C. Kesselman. The globus project: A status report. In *HCW '98: Proceedings of the Seventh Heterogeneous Computing Workshop*, page 4, Washington, DC, USA, 1998. IEEE Computer Society.

[6] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, New York, NY, USA, 1998. ACM Press.

[7] GAMA. GAMA: Grid Account Management Architecture. http://grid-devel.rocksclusters.org/gridsphere/gridsphere?cid=gama.

[8] GEON. GEON: The Geosciences Network. http://geongrid.org.

[9] GridAuth. GridAuth: Grid Authority. http://www.gridauth.com/.

[10] GridAuth. SourceForge: Project GridAuth. https://sourceforge.net/projects/gridauth/.

[11] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC Publication, January 1999. http://www.ietf.org/rfc/rfc2459.txt.

[12] B. S. Jr., T. Finholt, I. Foster, C. Kesselman, C. Beldica, J. Futrelle, S. Gullapalli, P. Hubbard, L. Liming, D. Marcusiu, L. Pearlman, C. Severance, and G. Yang. Neesgrid: A distributed collaboratory for advanced earthquake engineering experiment and simulation. In *13th World Conference on Earthquake Engineering (WCEE)*, August 2004.

[13] NCMIR. NCMIR: National Center for Microscopy and Imagine Research. http://ncmir.ucsd.edu/.

[14] NEES. NEES: Network for Earthquake Engineering Simulation. http://www.nees.org.

[15] nhdb. Neurosys hierarchical db. http://neurosys.cns.montana.edu/.

[16] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: Myproxy. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01)*, page 104, Washington, DC, USA, 2001. IEEE Computer Society.

[17] openssl. Openssl project. http://www.openssl.org.

[18] L. Pearlman, V. Welch, I. T. Foster, C. Kesselman, and S. Tuecke. The community authorization service: Status and future. *CoRR*, cs.SE/0306082, 2003.

[19] PURSe. PURSe: A Portal-based User Registration Service. http://www.grids-center.org/solutions/purse.

[20] SAML. Security assertion markup language (saml) working draft. http://www.oasis-open.org/committees/documents.php?wg_abbrev=security.

[21] B. Schneier. Sha1 broken. http://www.schneier.com/blog/archives/2005/02/sha1_broken.html.

[22] SRB. SRB: Storage Resource Broker. http://www.sdsc.edu/srb/.

[23] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. In *3rd Annual PKI R&D Workshop*, 2004.

[24] W.Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. Technical Report Internet RFC-1777, IETF, 1995. http://www.ietf.org/rfc/rfc1777.txt.