The GridSite Web/Grid security system

Andrew McNab*,†

*Correspondence to: Andrew McNab, Department of Physics, University of Manchester, Oxford Road, Manchester M13 9PL, U.K.
†E-mail: andrew.mcnab@man.ac.uk

SUMMARY

This paper describes the architecture of the GridSite system, which adds support for several Grid security protocols to the Apache Web server platform. These include the Globus GSI authentication system, Grid Access Control Language (GACL) access policy files, and Distinguished Name (DN) List and Virtual Organization Membership Service (VOMS) group memberships. Particular emphasis is placed on how the architecture of GridSite has evolved during the past 3 years, how this has been influenced by operational experience with production systems, and how the project has led to new developments, such as GACL. Finally, a description is given of how GridSite has been made to interoperate with other deployed security systems, both as producers and consumers of GridSite’s authorization information. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: GridSite; GSI; GACL; Apache; X.509

1. INTRODUCTION

GridSite [1,2] was originally developed as the management system for the GridPP project’s Web site [3]. The original architecture was a result of the special requirements of GridPP, and its need for security to span both Web and Grid environments.

The GridPP collaboration involves a community of about 100 particle physicists, computer scientists and site administrators. Most of the members are located at one of 20 particle physics groups at U.K. universities, or at the CERN (Geneva), SLAC (Stanford) or FNAL (Chicago) international laboratories, and are mostly either participants in running experiments at SLAC or FNAL, or in planned experiments at CERN. GridPP has developed Grid middleware as part of the EU DataGrid project [4] and is involved in the EGEE [5] and LHC Computing Grid [6] deployment projects.

These various affiliations create a need for overlapping domains of authorization, which correspond to software development subgroups, home institute or experiment association. Furthermore, different
areas of the Web site had to be to accessible members in one or more of these domains, with write access held by fewer members than read access.

Given our involvement in developing Grid middleware itself, it was natural to fulfill these Web site requirements using the Grid technologies. In addressing these requirements, a toolkit of new tools, GridSite, has been produced, which add support for Grid security protocols to the industry-standard Apache [7] Web server.

2. AUTHENTICATION

Most collaboratively maintained Web sites use some form of username/password authentication to identify users. This can readily be implemented using HTTP’s own password mechanisms (with both plaintext and MD5 hashed passwords) or with a combination of HTML forms and HTTP cookies which are created by the server, but stored and presented by the user’s Web browser.

However, all of the GridPP user community were being issued with X.509[‡] user certificates as part of the project’s Grid deployment, which offered an alternative authentication method.

These certificates were initially issued by the U.K. High Energy Physics Certification Authority at the Rutherford Appleton Laboratory. This Certification Authority (CA) was operated as part of GridPP to enable members to participate in the EU DataGrid and other projects, and was able to operate relatively quickly due to the small size and previous collaborations within this community.

The first version of the Web site software used a GSI-enabled version of the traditional Secure Shell system, produced by the Globus Project [8]. Each user who needed write access to the Web site was associated with a local Unix account on the Web server machine, and were able to upload and manipulate files using scp/ssh and their X.509 certificate.

Whilst this worked for the relatively small number of Web site authors at the time (up to about 12), it had obvious scaling problems in that Unix accounts had to be manually created and associated with certificate identities. Furthermore, most users felt that the Unix command line interface was not a natural way to maintain a Web site, which was most naturally interacted with via a Web browser.

During this period, some of the page formatting features of GridSite were developed (such as adding standard headers and footers to all pages) as runtime extensions to the Apache Web server. However, since our X.509 certificates were a technology shared between the Web and Grid worlds, we were able to go beyond using ssh and the Unix command line too.

When Netscape released its first browser with cryptographic security in 1994, authentication of both servers and clients was supported in their ‘Secure Sockets Layer’ (SSL) protocol. Since then, it has become commonplace for e-commerce Web sites to use ‘HTTP over SSL’§, and to present users’ browsers with a service certificate associated with the domain name of the server.

However, since the booming e-commerce industry was largely concerned with selling products and services to private individuals, the capacity to use client certificates has been largely ignored, in favour of using simple usernames and passwords to re-authenticate returning users.

‡X.503 version 3, IETF RFC 2459.
§HTTP over TLS, IETF RFC 2818.
Nevertheless, client-side support for cryptographic authentication using X.509 certificates remains in all major commercial and Open Source Web browsers. Coupled with the widespread issuing of X.509 certificates to GridPP members, this made it practical for us to use user certificates rather than usernames for everyday interactions with the GridPP Web site. In addition to the page formatting features, options were added to list directory contents, to upload or delete files, and to edit pages via HTML forms within the user’s Web browser. However, this required a security model to be developed to control the amount of read, listing and write access each user had. This architecture uses X.509 authentication, and the Virtual Organization (VO) groups and Grid Access Control Language (GACL) policies described below.

3. **VO Structure**

GridPP involves many working groups of several different classes—for example, middleware development groups, applications communities and deployment teams. An individual may be a member of several groups, and access control to areas of the Web site needs to reflect this structure.

However, every member has a unique X.509 certificate Distinguished Name (‘DN’) which can be used both for Web authentication and accessing our deployed Grid infrastructure. Consequently, the simplest way has been used to represent a group, by simply listing its members’ DNs. These ‘DN Lists’ are both stored internally and published as plain text files via HTTPS. They are also uniquely identified by their URI (which is an HTTPS URL) and access to them can be controlled using the GACL access policies described in the next section.

For example, https://www.gridpp.ac.uk/dn-lists/members lists all the members of GridPP, whereas https://www.gridpp.ac.uk/dn-lists/manchester lists only those members in the Manchester High Energy Physics group.

During the initial period of GridSite development, the major deployed system of Virtual Organization (VO) management was the VO-LDAP service developed by the EU DataGrid security group [9,10]. This publishes lists of user DNs within a VO or one of its subgroups, and these lists can themselves be identified by an LDAP URL. Other middleware developed by the EU DataGrid and by GridPP allows sites to use these lists for local access control. Since the output of these LDAP queries or a simple HTTPS fetch of our DN lists are identical, it is straightforward for a site to use either type of group information. However, for strict compatibility with the EU DataGrid system, a description is given below of a gateway to the GridSite DN Lists which uses the LDAP protocol.

4. **GACL Access Policies**

The groups defined by the DN Lists provide only half of the access control system. Some way of associating groups or individuals with rights to use or modify objects in the system is also required.

Non-Grid systems typically use some set of permissions associated with local usernames and groups on the resource itself. For example, Unix uses a bitmask of read, write and execute permissions, which give specific rights to the owner of the object (usually a file), the owner’s group or any other user of
the system; whereas the Andrew File System (AFS)\(^8\) uses a similar bitmask associated with Access Control Lists containing usernames defined by the site (the AFS Cell).

Since users of the GridSite system do not have Unix or AFS accounts on the server, the access policies are defined in terms of DNs of individuals or groups.

These policies are represented in an XML-based form, called GACL [1,2]. These GACL policies are composed of one or more Entry elements, which contain a list of permissions and a set of associated credentials. If a user has these credentials, then the permissions given in that Entry are granted. If more than one credential is stated, then they are all required for that Entry to apply. This permits the policy to refer to intersections between groups, where the intersection set does not have its own group in any VO. (For example, a policy might only give access to members of the University of Manchester who are also CERN users.)

Policies can also contain multiple entries, possibly with different associated sets of permissions. This allows different groups or individuals to be allocated different levels of access. (For example, to allow read permission to any user, but to reserve write permission for the Web site administrators.)

The basic set of permissions are Read, List, Write, Execute and Admin, and are orientated towards file and Web server applications. Permissions must be explicitly asserted by the policy for them to be granted. The Read, List and Write permissions allow reading, browsing and modifying files as one would expect. Execute is not used by the GridSite system itself, but may be given special meaning by other middleware, such as scripts which run on a GridSite enabled server. The Admin permission grants the right to modify the access policy itself.

The GridSite policy engine understands three basic credential types: a user DN, a DN List group (either with an HTTPS or LDAP URL), or groups or roles from VO Membership Service (VOMS) [9,10] attribute certificates.

Each policy must be associated with the resource it governs in some way. How this is done depends on the nature of the resource. For Web servers, GridSite provides a very simple way of associating GACL policies with parts of the URL-space and filesystem: the policy file governing access to a directory on the Web server is stored in a file named .gacl in that directory. If none is present, the .gacl file in the parent (or higher directories) is used.

The format of the GACL policies are considerably simpler than some of the alternatives, such as eXtensible Access Control Markup Language (XACML) [12], which are now beginning to be used in the Web services and Grid communities. GridSite is now being expanded to support XACML policies as an alternative to GACL, so that sites and users can make their own assessment of the potential interoperability gains of XACML versus the simplicity and ease of maintenance of GACL policies.

5. WEB MANAGEMENT INTERFACE

One of the requirements of the original GridSite design was to give users of any platform an intuitive interface to manage the Web pages and files stored on the server. This has been provided

\(^8\)Current AFS development is now largely the work of the OpenAFS Project [11].
by a management interface implemented as HTML forms, which allows authenticated users to create pages, edit text or HTML files, and to upload individual or collections of files—all without leaving their standard Web browser. All of these operations are controlled by the relevant GACL policies.

Additionally, the GACL policies themselves can be created or edited using an interface which represents the policies as HTML tables rather than raw XML; and an editor is provided for the DN Lists which enforces their simple format.

In the first version of GridSite to use HTTPS authentication, the management functions and access control were provided by a single binary using the Common Gateway Interface (CGI) to communicate with the Web server. Subsequent versions split this between the mod_gridsite Apache module, which provides access control and includes any appropriate management links in formatted HTML pages; and a smaller CGI program which provides the page, GACL and DN List editors, and the file and directory management functions.

6. OPERATIONAL EXPERIENCE

During 2001–2003, the GridPP Web server was operated using the CGI version of GridSite. This proved to be very successful, and many members took responsibility for managing their own areas of the site. Thus most of the content management responsibilities were removed from the ‘Webmaster’ team, who were able to concentrate on keeping the system operational, supporting user queries, and developing extensions to the software.

However, during this period several limitations of the architecture were identified, which impacted operational activities and administrative tasks. In particular, as the service became more popular, the performance implications of using CGI become a problem. The CGI interface requires that a new Unix process is spawned for each page or file request.

Additionally, GridSite did not take advantage of HTTP connection reuse (by outputting the HTTP Content-Length header for dynamic pages) and this was observed to cause particular performance problems with HTTPS: a new HTTPS connection involves a much bigger overhead than HTTP, due to the SSL handshake and the handling of cryptographic keypairs. With HTTP this is frequently ignored when writing CGI programs, but this is very inefficient for HTTPS.

In maintaining a production service with an increasing public profile, attention was paid to simplifying security maintenance of the server. This involved moving unrelated services to other machines, removing services that were not essential (for example, replacing anonymous ftp with HTTP) and establishing procedures to respond to software security vulnerabilities when announced.

One advantage of the GridSite CGI architecture was minimal dependency on changes in the Web server itself (Apache) [7], or the OpenSSL [13] cryptographic library used by Apache’s module which provides SSL/TLS support, mod_ssl. These were frequently updated in response to vulnerabilities, but no modification was required to GridSite at those times.

Finally, our approach of handling page and file requests via CGI prevented us from easily using other Web technologies, such as PHP or other CGI programs. As demand for these technologies increased from users (such as Ganglia’s [14] monitoring pages in PHP) ad hoc solutions were necessary which did not benefit from GridSite’s advanced security model.
7. MODULARIZED MOD_GridSite

During 2003 GridSite was redesigned to operate as a shared object module within the Apache 2.0 Web server, rather than as a standalone CGI program. This effectively made GridSite a fully fledged part of the Apache environment, and results in several operational advantages.

First, the GridSite handlers are part of every Apache process. Apache maintains a pool of these processes, which wait for incoming connections. Consequently, there is no additional process creation overhead when GridSite is involved in making an access control decision.

Second, this simplifies common administrative tasks, since GridSite can take advantage of Apache’s configuration and logging infrastructure, which site administrators are already familiar with.

Third, since GridSite access control decisions are taken within Apache, all of the dynamic content systems supported by Apache (PHP, mod_perl, JSP etc.) can be subject to the same GridSite security system.

In addition, it was ensured that Content-Length headers were properly output, so as to support connection reuse which considerably improves HTTPS performance.

This modularized system is now in production use on several sites, including the GridPP and LCG Grid Operations Centre sites.

8. GSI SUPPORT

Alongside the original CGI version of GridSite, experiments were carried out with mod_ssl-GSI [15], written by Mike Jones of Manchester Computing. This added support for Globus GSI proxies to the Apache 1.3 SSL handler, which then understood the non-standard certificate chain used by GSI to support delegation. These user credentials are of particular interest since they used our deployed Grid projects, and are the delegated credentials possessed by a remote batch job. Non-interactive Web clients with HTTPS support, such as curl [16] and wget [17], could then be used to carry out remote file transfers, using GSI proxies, within the job script.

On first using Apache 2.0 and developing mod_gridsite, an equivalent modification to the standard Apache 2.0 mod_ssl was written, which was distributed with beta versions of the modular GridSite. However, our experience with deploying beta versions showed this would be difficult to support, because of the frequent changes to the standard mod_ssl in response to security vulnerabilities.

For this reason, our GSI support was redesigned to exploit OpenSSL’s extensive use of callbacks, instead of patching the mod_ssl code. Functions were added to verify the special GSI certificate chains within mod_gridsite, and GridSite was make to dynamically modify the OpenSSL callbacks at runtime to use these functions.

In this way, strict dependency on Apache and OpenSSL versions were removed, so that new rebuilds of GridSite are not necessary each time an Apache or OpenSSL security vulnerability is found and corrected.

9. GridPP VO

One of the original aims of building the security system around the same X.509 user certificates used for the GridPP Grid deployment was to share authorization information between Web and Grid services.
The main production use of these capabilities has been to publish DN List groups for use by compute resources as input to authorization decisions.

The EU DataGrid project defined [9,10] a way of publishing similar VO information by LDAP. These VO LDAP servers have been maintained by deployment groups and applications, and define, for instance, the list of DNs of members of the Integration Team. Since GridPP was a major contributor to the EU DataGrid, both in terms of effort and compute resources, a VO LDAP interface was added to GridSite to allow our sites to access DN List information, managed via the GridSite Web interface. These lists are then downloaded via LDAP using standard EU DataGrid software and used to construct the local lists of approved users.

Towards the end of EU DataGrid, a new system, the Virtual Organization Membership System (VOMS) [9,10] was developed. This supported both a pull model similar to VO LDAP and based on HTTPS, and a push model in which the VOMS server issues short-lived attribute certificates to users, which they can present to resources to prove group membership. As with VO LDAP, VOMS HTTPS and attribute certificate interfaces have been added to GridSite, which give access to group memberships defined by DN Lists.

10. BaBar VO

As well as supporting DN List groups maintained by members of the GridPP collaboration, a VO service for the BaBarGrid project was also requested, for the BaBar experiment at SLAC. Membership of BaBar was already well defined by possession of an account on the SLAC Unix cluster with full BaBar privileges. Since their administrative system was largely designed to properly assign these privileges, it was natural to derive membership of their VO from the same source. This was implemented by allowing BaBar users to associate their certificate with their BaBar account at SLAC, and then by periodically importing a list of valid BaBar members into GridSite as a DN List. This was then available to the rest of GridSite, and permitted Web access decisions to be based on BaBar membership, and the list of members to be republished using the VO-LDAP and HTTPS as described above.

11. FURTHER WORK

Although the GridSite system’s use of X.509 user certificates for authentication is practical for its current user community, there is a need to support usernames and passwords as an alternative. In particular, ways to add support for Shibboleth [18] are being investigated.

In addition, support for the XACML [12] policy language is being added as a standards-based alternative to GACL.

As part of EGEE and LCG, Apache/GridSite is being extended to act as a hosting environment for Web and Grid services written in C/C++ and scripting languages.

REFERENCES

15. mod ssl-GSI. http://www.sve.man.ac.uk/Research/AtoZ/SAMD/Resources [31 May 2005].