WEB SERVERS FOR BULK FILE TRANSFER AND STORAGE

A. McNab, S. Kaushal, Yibiao Li, University of Manchester, UK

Abstract

GridSite has extended the industry-standard Apache webserver for use within Grid projects, both by adding support for Grid security credentials such as GSI and VOMS, and with the GridHTTP protocol for bulk file transfer via HTTP. We describe how GridHTTP combines the security model of X.509/HTTPS with the performance of Apache, in local and wide area bulk transfer applications. GridSite also supports file location within storage farms, and we explain how this has been implemented within Apache using the HTCP protocol, and the client-side commands and toolkit we have provided for applications.

INTRODUCTION

The GridSite Project[1] has developed a security toolkit, libgridsite, and a set of extensions to the Apache[2] web server to support Grid security credentials, authorization policies based on them, and read/write file operations.

Another paper[3] presented at this conference has summarised the Grid Credential processing and Access Control aspects of the GridSite Framework. In this paper, we describe how that framework can be used to provide bulk file storage with fine grained, virtualised access control, and how it has been extended to support HTTP transfers of large files with an unencrypted data stream, and third-party transfers of files between remote servers via HTTP(S).

FILE STORAGE

Web Servers such as Apache already provide access to files via the HTTP[4] protocol, either over unencrypted TCP or encrypted using TLS[5]. HTTP provides a GET method, which allows files, or byte-ranges of files, to be retrieved. GridSite's policy driven access rights model allows this to be limited to clients with the appropriate X.509[6], GSI[7] or VOMS[8] identity or group membership credentials.

The HTTP and WebDAV[9] specifications also describe methods which alter files on a server: PUT, DELETE and MOVE. Traditionally, these are not implemented by web servers, due to the lack of an appropriate access control model. GridSite's mod_gridsite shared object extension adds support for these methods directly to the Apache server, with rights covered by GridSite's policy engine described previously.

This combination allows the web server to be used as a read/write file server, equivalent to FTP, but using credentials relevant to Grids, and without the need to use underlying Unix accounts to control which users have access to which files: this is entirely determined by the policy files covering the directory or hierarchy of directories in question.

Since the combined Apache/GridSite servers are based on existing standards, they can be access in read or write mode using web clients from the mainstream web development community, such as the curl[10] command line tool.

To further simplify the use of these servers, we have also produced a family of command line tools, modelled on the scp command of OpenSSH[11]. htcp provides file copying to and from remote HTTP(S) servers; htrm allows remote files to be deleted; htmv provides renaming of files; and htls uses remote directory listings to display a list of files in the same format as the Unix ls command, with file size and datestamp information taken from the HTTP header values.

"GRIDHTTP"

The model described so far only permits access control on the basis of credentials supplied by an HTTPS request. This means that files will then be transmitted over encrypted TLS channels. For bulk file transfers, the encryption overhead may be inappropriate, and an uncrypted data stream is required.

To provide this we have developed the GridHTTP profile for using HTTP: this does not involve deviations from existing HTTP standards, but describes how to use existing headers and methods to produce an encrypted data stream.

First, an HTTPS request is made for the file, with an Upgrade: header present. This includes the protocol string GridHTTP/1.0, and alerts GridSite servers that an unencrypted data stream is required.

If the client has the correct rights to access the file in question, the server issues an HTTP redirect from the HTTPS URL to an HTTP URL (corresponding to an unencrypted channel.) The redirect response includes a HTTP cookie, which a onetime passcode which the client must supply to obtain the file over HTTP.

The client obeys the redirection (which is the default behaviour for htcp, and is readily achieved with a command-line option in curl) and presents the onetime passcode to gain access.

When the server receives the passcode over an unencrypted stream, the passcode is marked as no longer valid and it cannot be used for subsequent requests.

This model shares the vulnerability to “man in the middle” attacks which all unencrypted data stream protocols face (such as The Globus Project's[12] GridFTP), but is resistant to replay attacks due to the onetime nature of the passcode.

THIRD-PARTY TRANSFERS

When managing bulk files it is frequently necessary to transfer files between remote servers, but it is inefficient to do this via the machine which is orchestrating the transfer. (For example, if the management machine is a user's notebook computer on a poor network connection.)

Consequently, systems such as GridFTP support so-called “third party transfers” between remote sites, with control channels between the management machine and the data servers, and a data channel for the transfer of the large files themselves.

The WebDAV HTTP specification also defines a COPY method which can transfer files between two remote URLs, and we have implemented this as a CGI program supplied with GridSite, gridsite-copy.cgi. When presented with a copy request from the management machine, the gridsite-copy.cgi program on one of the data servers attempts to fetch the remote file via GET and place it on its own file space. Both GET and the local file operations are controlled by GridSite's fine grained access control model.

However, in doing this, the gridsite-copy.cgi instance must somehow be associated with the user who is orchestrating the transfer, and be able to prove this to the data server with the file. Some form of delegation is needed.

Previously, we have investigated using GSI proxy delegation for this purpose, but following the development of our GridHTTP profile, with its onetime passcodes, we have implemented gridsite-copy.cgi using a much simpler authentication process.

To perform a third-party transfer, the user's management machine makes a GridHTTP request to the data server with the file, as described above, using HTTPS and supplying their Grid credentials (X.509 certificate or GSI proxy, plus VOMS attriutes if present.) Instead of following the redirect to the HTTP server itself, the management machine then issues a COPY request to the other data server, using the onetime passcode it has received and the HTTP URL. Since this COPY is also transmitted as an encrypted HTTPS request, the onetime passcode is still not revealed, and can be used again by the gridsite-copy.cgi program which receives it. (In this way, we define “one time use” as one use of the passcode over an unencrypted HTTP channel.) gridsite-copy.cgi can then issue the GET itself, using the passcode, as if it had made the initial GridHTTP request.

Again, because this protocol is merely a profile based on existing standards, it can be used by standard command line clients such as curl, if the correct arguments are given. For simplicity, it is also supported by our htcp command line tool.

SITECAST FILE LOCATION

As well as the transferring large files, it is also necessary to locate them. As more large are deployed with the majority of their disk space on the CPU nodes, such as the Tier-2 facility at the University of Manchester [13], the file location problem is becoming similar to the problem of finding files on farms of web servers or web caches.

With this in mind, we have adapted the Hypertext Caching Protocol[14] (HTCP – not to be confused with the GridSite htcp command) to provide a file location system.

HTCP queries are short, binary messages, which can readily be transmitted over UDP. By using UDP multicast, we are able to query all data servers in parallel.

We have implemented a UDP responder as part of the mod_gridsite Apache module, and this simply queries the underlying Unix filesystem to determine the presence or absence of the file. Using a 3GHz Pentium IV processor machine, we are able to receive replies within less than 1ms of elapsed time, due to the lightweight nature of UDP and the simplicity of the filesystem query (a call to the Unix stat() function.)

The client side of this SiteCast system is implemented as part of htcp, and allows files to be copied from virtual site-wide URLs to the local machine: htcp makes a SiteCast query to find one or more replicas of the file, picks the first host to respond (which is likely to be the least loaded) and copies the file from there to the user's filesystem.

Further additions are planned to this system, including a POSIX interface for clients, and an SRM interface to allow external queries to made to the site about the presence and location of files.

A significant advantage of the system is that no central database of all files is needed, and there is no danger of such a database becoming out of synchronisation with the files present on data servers, as GridSite creates new copies of files atomically (by writing to a temporary file and then renaming the file atomically when the transfer is finished.) Furthermore, offline data servers automatically remove their replicas of files from then location system, by simply not being able to respond to SiteCast queries. Finally, IP multicast messages can be limited to groups of machines, which machines subscribe to, and other groups can frequently be filtered out at the lower levels of network cards and their drivers. Many IP routers also allow filtering and routing of multicast packets, and this may be used to partition sites, or join sites together to form virtual data server pools.

CONCLUSION

GridSite provides a flexible framework for managing access to web servers using common Grid credentials. This system can now be used for providing secured read/write file servers, with the option to transfer files via unencrypted data streams and by third-party transfers between remote GridSite servers.

The SiteCast protocol allows files to be located within a site, without the need for heavyweight databases or headnodes, since the work of responding to the query is carried out in parallel by all machines within the relevant multicast groups.

ACKNOWLEDGEMENTS

This work was funded by the UK Particle Physics and Astronomy Research Council, as part of the GridPP project and the e-Science Studentships programme.

The design of the GridHTTP and SiteCast systems benefited greatly from discussions with Dr M. Jones and Dr A. Forti, both of the University of Mancheser.

REFERENCES

[1] http://www.gridsite.org/.

[2] http://www.apache.org/.

[3] A. McNab et al., "Web Services with GridSite and C/C++/Scripts", Proceeding of CHEP 2006, Mumbai, India, 2006.

[4] IETF RFC 2616, "The HTTP Protocol", http://www.ietf.org/rfc/rfc2616.txt

[5] IETF RFC 2246, "The TLS Protocol", http://www.ietf.org/rfc/rfc2246.txt

[6] IETF RFC 3280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", http://www.ietf.org/rfc/rfc3280.txt

[7] IETF RFC 3820, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile", http://www.ietf.org/rfc/rfc3820.txt

[8] R. Alfieri et al., Managing Dynamic User Communities in a Grid of Autonomous Resources, TUBT005 - Proceedings of CHEP 2003, 2003.

[9] IETF RFC 2518, "HTTP Extensions of Distributed Authoring – WebDAV", http://www.ietf.org/rfc/rfc2518.txt

[10] The Curl project, http://curl.haxx.se/

[11] The OpenSSH project, http://www.openssh.com/

[12] The Globus Project, http://www.globus.org/

[13] A. Forti and A. McNab, "Cluster distributed dynamic storage", Proceedings of CHEP 2006, Mumbai, India, 2006.

[14] IETF RFC 2756, "The Hypertext Caching Protocol", http://www.ietf.org/rfc/rfc2756.txt