WEB SERVICES WITH GRIDSITE AND C/C++/SCRIPTS

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

Abstract

GridSite provides a Web Service hosting framework for services written as native executables (eg in C/C++) or scripting languages (such as Perl and Python.) These languages are of particular relevance to HEP applications, which typically have large investments of code and expertise in C++ and scripting languages.

We describe the Grid-based authentication and authorization environment that GridSite provides, removing the need for services to manipulate Grid credentials (such as X.509, GSI and VOMS) themselves. We explain how the GRACE model (GridSite - Apache - CGI - Executables) allows Unix-account sandboxing of services, and allows sites to provide hosting of multiple services provided by third-parties (such as HEP experimental collaborations) on the same server. Finally, we propose scenarios which combine GridSite's authorization model with service sandboxing to allow remote deployment of services.

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.

In this paper, we describe how that framework can be used to build Web Services, which are protected by Grid credenitals and access policies, and which can be written in any of the languages supported by Apache's Common Gateway Interface (CGI) – that is, binary executables derived from C or C++, and all of the major scripting languages, including Perl and Python.

Combined with a service sandboxing model we have developed using temporary Unix accounts, we refer to this Web Services framework as GRACE, for GridSite – Apache – CGI – Executables.

GRID CREDENTIALS

The GridSite extensions to Apache take the form of a loadable module, mod_gridsite, which is dynamically linked to the Apache executable at server startup time. This model gives full access to all of the Apache's internal data structures, and in particular to the mod_ssl module which processs the SSL/TLS encryption layer of HTTPS connections.

This is of particular important in Grid applications, where clients authenticate using X.509[4] certificates, either as conventional certificates possessed by users and issued by Certificate Authorities, or as GSI Proxy Certificates[5], created by users and given to agents or remote jobs.

Apache's mod_ssl correctly extracts the user identify – the Distinguished Name or DN – from standard X.509 certificates, and mod_gridsite adds support for GSI Proxies. This is done by dynamically modifying the SSL callbacks, to wrap the standard processing in mod_gridsite functions which can parse GSI Proxy certificates. This stage is also used to look for the presence of VOMS[6] attribute certificates.

Once mod_gridsite has these credentials parsed and verified using their cryptographic signatures, they are made available to Apache and CGI programs as environment variables. This follows the CGI convention for transmitting “out of band” information to CGI programs in the Unix process environment.

ACCESS CONTROL

mod_gridsite provides an internal access policy evaluation engine, and both the results of this evaluation and the raw credential values are available to CGI services for any additional fine grained access control they wish to apply.

Policies are specified in GridSite's XML-based Grid Access Control Language, or in an equivalent subset of XACML[7]. They can require clients to possess a full X.509 certificate or GSI proxy with a specific DN; a VOMS attribute certificate with specific groups, roles or capabilities; that the client's DN is a member of a given DN List group, downloaded asynchronously and cached, from a VO-LDAP or VOMS HTTP server; or that the client has a specific IP address or subnet.

Policies can involve multiple credential requirements (that is, a logical AND), or alternative credential requirements (a logical OR.) Permissions may be granted but also denied on the basis of the requirements specified (a logical NOT.)

The default set of permissions (read, list, write, execute and admin) are appropriate for file access and the exeuction of CGI programs on web servers.

CGI EXECUTION

Using the Apache/GridSite framework, Web Services are implemented as binary executables or scripts, and the request and response are communicated via the CGI protocol, using the Unix stdin and stdout of the process.

By default, Apache will execute CGI programs as the Unix user the server process is running as, such as “apache”, which will typically have write access to many files associated with the web server, including executables. This imposes a very coarse-grained security policy, as anyone with access to the apache account, including the ability to supply CGI executables, can alter any of the web server files.

Apache provides a mechanism for running CGI programs as different Unix users, using a setuid binary, suexec. In this case, the CGI program is run as the user who owns it, and it is relatively straightforward to manually administer a small number of users and grant them Unix command-line or ftp access to the server.

Although this provides for a degree of separation between owners of different groups of files and servers, it is not sufficiently flexible to implement the Grid ideal of virtualised use of resources, which can be used by users who have no previous association with the resource, and whose access rights are defined dynamically.

GSEXEC

To address this shortcoming, we have developed a replacement for suexec, called gsexec. This is backwards-compatible with suexec, but has additional configuration options which can be set via the Apache configuration file.

Gsexec runs CGI programs as Unix users which are different from the main apache user account, but instead of being static accounts which have to be administered manually, temporary accounts are allocated from a pool of Unix users dedicated to the gsexec system.

Gsexec has two modes of operation, depending on how pool accounts are associated with a given request. Either the pool account is allocated to the client's certificate DN when it is first seen by the server (unless the account is found to be no longer used, and then recycled) – in this mode, subsequent requests from the same client are allocated to the same pool account; or the pool account is allocated on the basis of the Unix filesystem directory which corresponds to the URL of the request.

In the first mode, the Unix filesystem and process permissions prevent sessions associated with different clients from interfering with each other: even if CGI services would otherwise allows different clients to invade each other's privacy or damage each other's files or shared memory segments, then Unix permissions will prevent that vulnerability in this mode.

In the second mode, the service author may maintain shared files, shared memory segments and database sessions which are accessible irrespective of the client making the request. This mode not only allows clients to access the service without manual granting of access rights – they can, for instance, have access rights as members of a group – but also allows service owners to be granted write access to CGI directories on the web server, purely on the basis of access policies. An unused Unix pool account will then be allocated to the group of services and files in that directory.

This mode allows the remote deployment of services by service owners, using the GridSite file server features to upload the service, and then gsexec to execute the service executable or script.

SOAP AND WEB SERVICES

CGI scripts or executables must accept and generate SOAP messages to qualify as Web Services. Libraries and modules to provide this are available for all of the major programming languages.

The GridSite distribution includes one Web Service, a GSI delegation service, and this is a useful example of how Web Services can be implemented in the Apache/GridSite framework: the gridsite-delegation.cgi service is written in C, and uses the gSOAP[8] package to provide parsing of SOAP messages, and to generate the code framework from the WSDL specification of the service.

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 Web Services, with multiple client sessions or multiple service owners. In these scenarios, Unix account permissions are used to isolate different authorization domains. Crucially, all of these access rights are determined dynamically, using policy files and pools of temporary Unix accounts.

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.

REFERENCES

[1] http://www.gridsite.org/.
[2] http://www.apache.org/.
[3] IETF RFC 2246, “The TLS Protocol”, http://www.ietf.org/rfc/rfc2246.txt
[4] IETF RFC 3280, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, http://www.ietf.org/rfc/rfc3280.txt
[5] IETF RFC 3820, “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”, http://www.ietf.org/rfc/rfc3820.txt
[6] R. Alfieri et al., Managing Dynamic User Communities in a Grid of Autonomous Resources, TUBT005 - Proceedings of CHEP 2003, 2003.
[7] The XACML Committee, http://www.oasis-open.org/committees/xacml/
[8] The gSOAP framework, http://www.cs.fsu.edu/~engelen/soap.html