Practical access control using NDG security

Practical access control using NDG security

Bryan N Lawrence[1], **Philip Kershaw**[1], Jon Blower[2]

[1]NCAS/British Atmospheric Data Centre, Rutherford Appleton Laboratory, STFC
[2]Reading e-Science Centre, Environmental Systems Science Centre, University of Reading

Abstract

Access control in the NERC DataGrid (NDG) is accomplished using a combination of WS-Security to ensure message level integrity, X.509 proxy certificates to assert identity, and bespoke XML tokens to handle authorization. Access control decisions are handled by Gatekeepers and mediated by Attribute Authorities. The design of the NDG-security reflects the reality of building a deployable access control system which respects pre-existing user databases of thousands of individuals who could not be asked to reregister using a new system, and pre-existing services that need to be modified to take advantage of the new security tooling. NDG-security has been built in such a way that it should be able to evolve towards the use of community standards (such as SAML and Shibboleth) as they become more prevalent and best practice becomes clearer. This paper describes NDG-security in detail, and presents experiences deploying NDG-security both in the e-Science funded NDG and the DTI funded Delivering Environmental Web Services (DEWS) projects. Plans for the future of NDG-security are outlined; both in terms of application modification and the evolution of NDG-security itself.

1. Introduction

The NERC DataGrid (NDG), which originally consisted primarily of a partnership between the British Atmospheric Data Centre (BADC) and British Oceanographic Data Centre (BODC), along with the STFC e-Science department, was established as a NERC funded e-Science project with the aim of improving integration of data holdings in the atmospheric and oceanographic sciences to the point where users could mix and match data from multiple sources in a "single-sign-on" context [1]. The NDG is now nearing the end of a second tranche of funding, with partners extended to include the Plymouth Marine Laboratory and the National Oceanography Centre, Southampton. The mandate has also been extended to look at integrating data holdings on a wider scale, although the NDG is still only committed to delivering integration of some classes of atmospheric and oceanographic data.

There is a common assumption in the environmental sciences community that access control on data is something that is not necessary and positively gets in the way of "doing science". Many academics have asserted (and continue to assert) that the first thing the NDG should have done was to throw away the single-sign on requirement in favour of uncontrolled access. None of those academics has the responsibility of providing services that are used by a significant number of people, or that deliver a significant amount of data. Experience tells us that it doesn't take many users before I/O subsystems are completely overloaded (never mind CPUs or networks). In such an environment, it is crucial to have methods to constrain access to priority users. When a data provider is also holding data which is either owned by, or for which the IPR resides with, third parties, there is also a requirement to protect the legal rights of those third parties (whether or not some individuals want it done at the time, their employing organizations may well take a different position at a later date). Sometimes too, data providers may be holding data that is in an interim form that is not suitable for general use; in this case too access needs to be limited. There are thus two driving requirements for access control: to protect access to finite resources, and to ensure that the use of those resources is appropriate (legally and/or scientifically).

The remainder of this paper describes the access control mechanism developed for the NDG, and its application in both the NDG and the Delivering Environmental Web Services (DEWS) projects. DEWS is a technology demonstrator project, that is using the same underlying technology to (1) take marine forecasts from the Met Office and deliver them into Search and Rescue applications at British Maritime Technology, and (2) take primary data from the health sector (e.g. GP admissions etc) and feed them into a system which returns predictions of hospital admissions broken down into geographical areas. (The prediction systems themselves are not part of the DEWS project; it

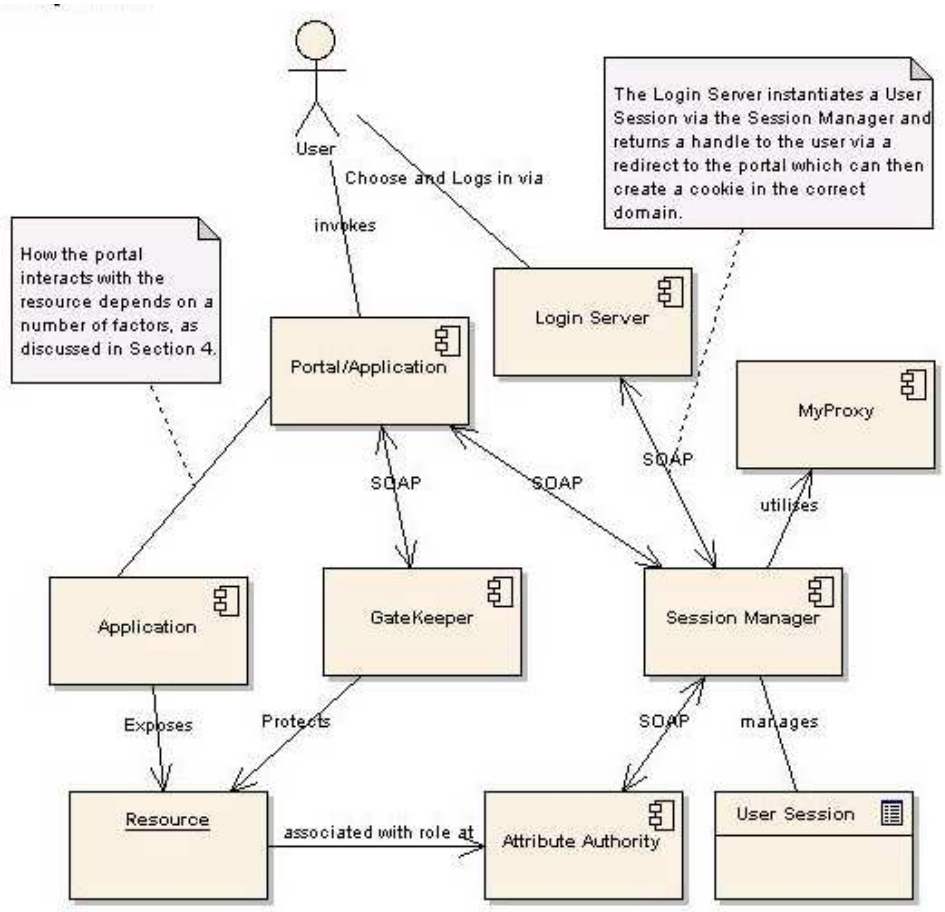

Figure 1: Key Components in Security Transactions

is the web services and data standards that share the same technology heritage.)

1.1 Security Requirements

Early in the design of the NDG activities it became clear that a number of guiding principles were going to affect what could be accomplished:

1. Data holders could not, and would not, consider changing the way they stored data and fundamental metadata (including user information).
2. User information could not be shared between partners without explicit authorization by the users on a case-by-case basis.
3. Real existing licensing constraints would affect what common "authorization roles" could be established, and that with more than two partners, common roles across all partners would be few.
4. Real existing users could not be asked to work directly with X.509 certificates.
5. Any new access paradigm had to support what was already possible and did not necessarily have to be any more secure than current practice (although all recognized better security would be good).
6. The technology used had to be portable and easily deployed in groups that have little resource for "tinkering". (Often such groups have little or no Java experience.)

At the time (2002/03), there was no existing access control paradigm that addressed all these principles satisfactorily, and so the NDG team reluctantly took a decision to engineer their own! Given the resource available a further principle emerged:

7. NDG security had better be simple to devise and build, as well as easy to deploy and use.

2. NDG-Security

NDG security provides access control by ensuring that users are granted permission to access a resource by a "gatekeeper", which has access to (1) an authentication token produced by means of an X.509 certificate issued by a "Login-Server" at one of the partners, and (2) an authorization token issued by the "Attribute Authority" associated by the gatekeeper with the resource in question. At that level the NDG-security mechanism is not fundamentally different from other solutions in this space.

2.1 NDG Authorisation

NDG uses a simple non-hierarchical role based access control mechanism. Roles are issued for users in Attribute Certificates (authorization tokens) issued and digitally signed by Attribute Authorities ("AAuthorities"). The current format is a bespoke XML document, it is highly likely that the next version of NDG will use a SAML [2] format token. The access control policy for a resource is based on a simple triple: for a Resource A, access control depends on Role B known by AAuthority C. Any gatekeeper that is presented with an attribute certificate obtained from AAuthority C that asserts that the user has role B can then grant access to A.

Principles 3 and 7 listed above led the NDG to decide on a bilateral trust mechanism. This is realized in the detail of how AAuthorities issue Attribute Certificates. The bilateral step comes from including within each AAuthority the mechanism to establish (zero, one or more) mappings between local roles and those known by remote AAuthorities. Thus, if AAuthority C trusts AAuthority D, and C has an equivalence mapping between role B (known by C) and role E (known by D), then AAuthority C can issue a "mapped attribute certificate" to a user (or their software agent) in response to an attribute certificate from D with role E. This "mapped attribute certificate" is an assertion from C that the user has role B, and can thus be used by the gatekeeper process to authorize access.

The role mapping is expressed at each AAuthority by means of an XML based role-mapping file. This contains details of which organizations the AAuthority trusts and for each of these what role mappings apply. This could be thought of as equivalent to an attribute assignment policy [3].

It is currently a design decision that only one level of mapping is allowed, since the NDG participants could then only envisage a situation where trust could only reliably be extended after an actual human bilateral relationship between the data providers (in practice many of the roles are intended to reflect real legally binding contracts by the users as to what they can do with the data and so the initial role mapping has to be done by a human).

2.2 NDG Authentication

Authenticating users across multiple domains is now a common problem, and there are multiple solutions: ranging from Shibboleth [4] assertions to OpenID [5]. NDG envisages browser and non-browser based profiles. The browser based profile is treated in detail here. In this case, the NDG mechanism is not fundamentally different from Shibboleth or OpenID.

NDG users start by logging in, usually in response to a request to access to restricted data in the situation where the portal doesn't yet have access to the required user credentials to present to the gatekeeper.

The key components in the process are shown in Figure 1. In this case, for browser access to a portal, the gatekeeper process will launch a login page that has much the same functionality as a Shibboleth WAYF (Where Are You From). In the NDG case, the options presented will be for "Login Servers" associated with the AAuthorities that are listed in the mappings available to the AAuthority protecting the resource in question. So for example, continuing with the previous nomenclature: Should a user be trying to access resource A, the login page would allow the user to choose between two login servers: those associated with AAuthorities C and D.

Each Login Service utilizes a MyProxy [6] server via its own Session Manager service. In NDG that MyProxy server is populated by lightweight[1] certificates issued by a SimpleCA, itself populated and updated directly from existing user databases, but more reliable certificates could be used.

On response to a login request, the Login Server requests a "Session Manager" to authenticate the user against their MyProxy credentials and create a "User Session" with a "Credential Wallet" within which the users proxy certificate [7] is held. The Session Manager returns the Login Server a handle to the User Session it has created. In the case of a browser process, the handle is then returned via a redirect to an HTTPS GET to the original application (the address of which is passed in the redirect to the Login Server). The application can then populate a cookie with the encrypted Session Manager address and User Session details for subsequent use.

2.3 Obtaining Credentials

Requests from the portal to access the resource are accompanied by the (Session Manager, User Session handle) tuple. The gatekeeper can then request the Session Manager for a specific attribute certificate associated with the appropriate AAuthority.

[1] Lightweight: no real effort has been made to secure the SimpleCA.

If the user session does not hold an attribute certificate from the AAuthority associated by the gatekeeper with the resource, then the Session Manager can use the proxy certificate to request an attribute certificate from the AAuthority. If the user is not known by that AAuthority, the session manger can obtain a list of *trusted* AAuthorities for that resource, approach them with the user proxy, take that remote attribute certificate and use it to obtain a mapped attribute certificate from the original AAuthority.

A walk through of the entire sequence from login through to a Gatekeeper decision based on a mapped an attribute certificate from AAuthority C (based on Authority D) is show in Figure 2.

2.4 Known Exploitation Stategies

One of the strengths of the NDG-security mechanisms is that apart from applications which have obtained individual credentials, the Session Manager is the only entity with access to credentials which needs to be accessible outside a firewall: the MyProxy Servers do not need to be visible outside. This means that all

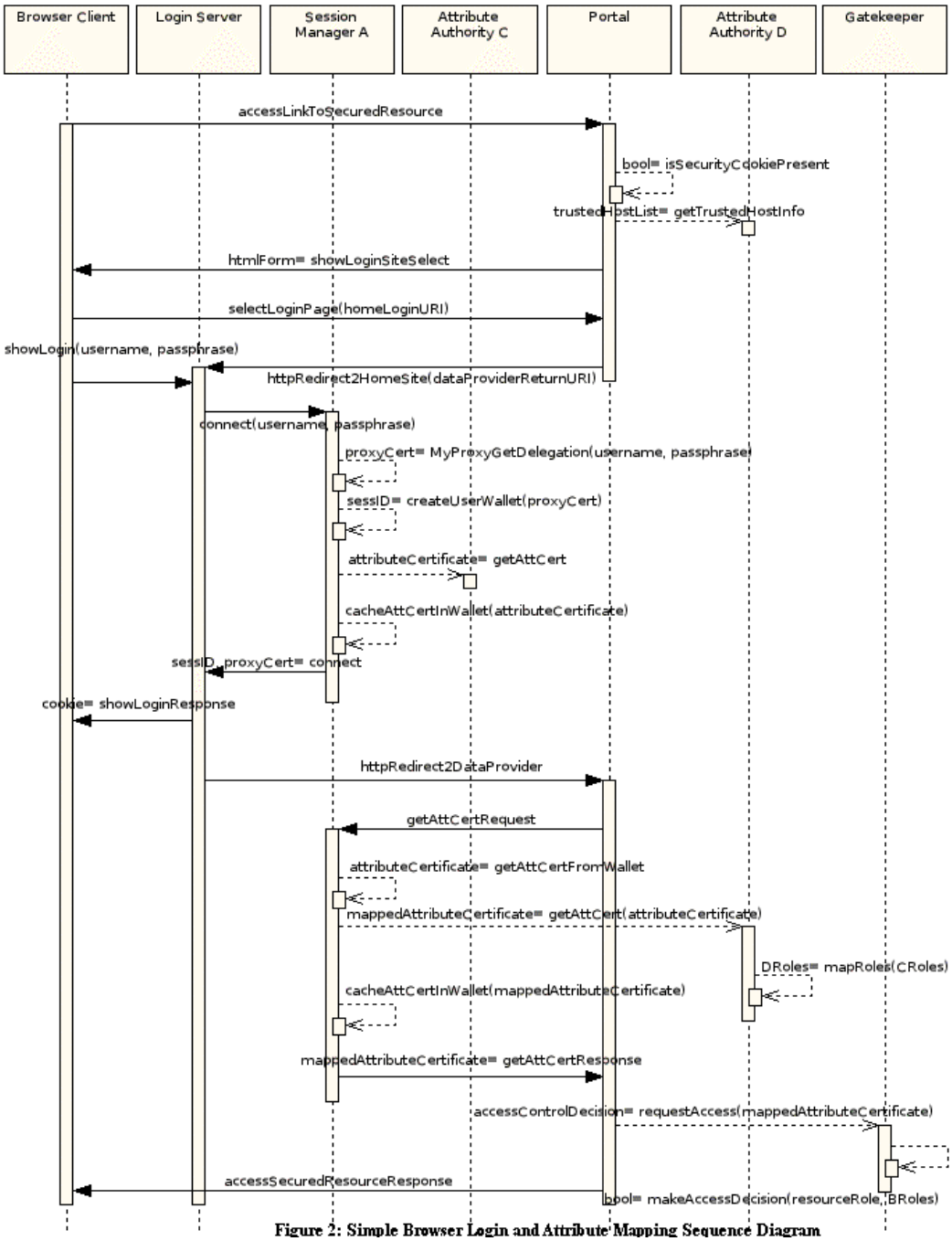

Figure 2: Simple Browser Login and Attribute Mapping Sequence Diagram

exploits are limited to obtaining proxy and/or attribute certificates at best.

There is, however, an obvious opportunity for a form of phishing in NDG-security: anyone could produce a "rogue" service which when a user attempts to access it passes a request to a "proper" Login Server. That rogue service, exploiting the open access NDG source code, could then get access to the use of a proxy certificate via the redirect holding the Session Manager handle. The rogue could then purport to require access to a "rogue Attribute Authority", at which point the Session Manager would give up the proxy certificate to that rogue in an attempt to obtain an attribute certificate. While the NDG currently sees this as a low risk, the most obvious way to avoid the problem is to give all login servers the capability to register associated portals (and/or applications), and require redirections to be signed. This has not yet been implemented.

2.5 Non-Browser Based Profile

Provided the client holds a valid proxy certificate it can be used to obtain the required attributes from an AAuthority where the user is registered by sending a request signed with the proxy. The client can manage and cache attribute certificate credentials locally in its own Credential Wallet. Alternatively, it can delegate this task to a Session Manager. The client authenticates with the Session Manager and its User Session and Credential Wallet are held there remotely.

2.6 Software Implementation

The core NDG security software consists of two components: the server suite, needed to deploy an AAuthority etc, and the client suite, used to build applications.

Both components have been been engineered in Python and have been recently rebuilt as Python eggs to minimize problems with dependencies. The server component currently uses Twisted[2] as the execution environment. The client egg provides a Session Manager and AAuthority client interfaces that can be plumbed into existing applications, and NDG-security middleware exploiting that client in a WSGI[3] package is nearing completion.

The Session Manager and AAuthority services use SOAP based interfaces with WS-Security [8] digital signature and transport level encryption to secure messages.

3. NDG-security in DEWS

As outlined in the introduction, the NDG software has been deployed in DEWS, which introduced a new set of requirements over and above those in the original NDG project.

Within DEWS, the major requirements were to (1) streamline the security for a case where the data transfers between one organisation and the servers on the boundary of another were to be controlled by NDG-security, but "the final mile" was internal to the consumer organisation; and (2), deploy NDG-security to protect a specific implementation of the OGC-Web Coverage Service (WCS) [9]. While this latter case is definitely an NDG use-case, the NDG experience had thus far been on securing Python applications that could be heavily modified by including the NDG-client code directly.

In both DEWS cases the consumers of NDG-security are Java applications, but using differing execution environments. While the project set out to use IBM WebSphere exclusively, various practical issues with application portability led to a different strategy. Now IBM WebSphere is used as the container for the gatekeepers and for the portal, and Apache WSS4J[4] is used as an additional client for the Marine stream. In addition, the DEWS-WCS itself is run in Apache Tomcat[5].

Using the native filtering capability of the containers to do the WS-Security, and having them interact with the NDG-security Python server implementations of WS-Security has been an interesting exercise in the real state of interoperability in this area. In practice, the actual configuration of what, and how, blocks in the messages are signed was a major stumbling block with progress held up by weeks on rather trivial steps. One further unexpected difficulty was also discovered with establishing the chain of trust for proxy certificates: because the standard web service containers expect to have static key stores it is difficult to set up signature verification with dynamically generated certificates.

In DEWS, one of the "users" is in fact a downstream server platform (BMT SeaInfo), which performs search and rescue predictions, thereby adding value to the data served by DEWS-WCS. As was expected, modelling SeaInfo as a dedicated "user" with a static certificate was easy to setup with NDG-security, however, more complicated problems were

[2] twistedmatrix.com
[3] www.wsgi.org
[4] http://ws.apache.org/wss4j/
[5] http://tomcat.apache.org/

encountered with the WCS access, primarily because of the data volumes.

4. Large Data Issues

One of the major issues faced is how to secure access to both applications and the data they deliver. While the NDG-security paradigm can deal with the logical issues associated with making a policy decision, the practical issue still remains how to deal with the transactions themselves, and in particular key transactions that result in large data transfers (potentially of gigabytes to terabytes).

This issue manifests itself directly when dealing with the OGC WCS specification in the context of both the marine data server in DEWS and the wider NDG. This specification imposes no limit on the size of data that can be requested. That leads to a number of possible responses to a data request, not all of which are supported by the vanilla WCS specification which was not produced with large met-ocean data transfers in mind.

The WCS specification describes a Web Service that receives requests for data either as a URL (via HTTP GET) or an XML document (via HTTP POST with or without SOAP). If security were not required, there would be no problem with delivering large datasets.

To protect the WCS, the DEWS gatekeeper web service acts as a proxy to it intercepting data requests and checking their security credentials. In this respect it is similar to the model used in the recent OGC GeoDRM interoperability experiment[6]. Communication is secured using WS-Security with transport level encryption as required (HTTPS), and is implemented in WebSphere. Two gatekeepers are deployed in DEWS, one each for the health and marine streams. The code is identical: the Gatekeepers know nothing about the data server that they protect. They simply check the security credentials of the clients' requests and forward the unmodified requests to the underlying data server.

In order to use WS-Security, all data requests must be formatted as SOAP messages that contain the required security credentials. The current Gatekeeper is a typical SOAP Web Service in that it receives SOAP messages and responds with SOAP messages ("SOAP in SOAP out"). This is fine for the Health stream of DEWS (in which the data sizes are small enough to be encoded in SOAP) and for most messaging in the Marine stream of DEWS. However, because the WCS allows arbitrarily large file responses, and because in the Marine stream large files are actually needed, a problem arises: such files are too big to be encoded in XML SOAP messages, yet the Gatekeeper can only respond to clients with SOAP messages (SOAP "out").

When one does not wish to modify the application code stack, there are three possible solutions, (1) Stream binary data through the gatekeeper; (2) Use SOAP extensions for binary data transfer, and; (3) Construct a new data delivery web service to wrap the original service with additional new access control syntax.

The first of these solutions would involve changing the Gatekeeper so that it is able to stream binary data (i.e. NetCDF files) as a direct response to a SOAP message. That is to say, change the Gatekeeper from purely "SOAP in", "SOAP out" behaviour to allow "SOAP in", "data out" behaviour for the downloading of data. However this would break all the available software tooling, require a lot of new code, and result in a particularly unportable solution.

The second option would exploit one of three binary transfer extensions to the SOAP standards. The impracticality of encoding large binary datasets as XML in SOAP is widely acknowledged, and has resulted in (at least) three options: (1) Base-64 encoding , (2) SOAP with Attachments (SwA), and (3) Message Transmission Optimization Mechanism (MTOM). Of these three, vanilla base-64 encoding is not really a solution for large datasets. The second two are not conceptually significantly different. Some tests were carried out with a WebSphere based solution using SwA but ultimately, for the large data transfers required this would not provide a solution.

The third option, like the first, results in a bespoke solution, but the methodology is relatively easily extensible for modification into a variety of other applications. The existing DEWS prototype adopts this approach, handling the access control requests using the Gatekeeper, resulting with the final data request returning an obscured URL and a unique job identifier. To access the data, the client digitally signs the identifier with its private key and passes the identifier and the signature in the URL to retrieve the data. On receipt of the request, the data server checks the signature against a public key contained in its key store and returns the data. The signature ensures that the token is of no use to an attacker without an appropriate private key.

[6]Details are available via OGC members.

While all these mechanisms require an NDG-security aware client, none of them require modification of the original application. If access to the application code stack is also available, then the access control can be done by SOAP messages independently of the data access and the problem does not exist. However, the third mechanism provides much of the same code and syntax that would be used, in particular, the token exchange could be identical, but with an internal gatekeeper process communicating with attribute authorities etc.

5. Discussion

One key area where NDG differs from other security solutions is the use of role mapping. The discussion here focuses on this area of the security model.

The role mapping function is performed by AAuthorities with current implementations in both DEWS and NDG having the role mapping functionality independent of the gatekeeper[7], however it would not be a major step to merge that functionality into a gatekeeper. In such a case, the (standard, non-federated) mapped attribute certificate would no longer have to actually exist as a document in its own right, although the conceptual function would remain. This would simplify the process of obtaining attributes since the additional step of obtaining a mapped attribute certificate would be eliminated. Even so, with the existing approach, this problem is mitigated to an extent by the fact that attribute certificates obtain from previous access requests are cached in a user's Credential Wallet.

Furthermore, there are use cases where the independence of role mapping from gatekeeper is useful: not all data providers will want to run Attribute Authorities, but they might want to exploit a third party Attribute Authority. This provides a useful level of functionality for real virtual organizations: For example, the constituent universities which make up the National Centre for Atmospheric Science (NCAS) may want to run Gatekeepers locally which bind roles understood by an NCAS Attribute Authority deployed at the BADC to resources which they deploy locally.

From a client side perspective, in the process to broker access to a resource the additional step of obtaining a mapped Attribute Certificate adds complexity. This could be exacerbated in the case where a request covers resources across multiple organizations.

Another consideration in the use of role mappings in the security architecture is their long-term maintenance and their use in trust relationships. If one party changes or deletes a role name the trust relationship is broken.

Role mapping has the major benefit of enabling data providers to share their existing data with other organizations without the need to change the associated role names. Even so, there may be exceptions to Principle 3 outlined earlier. Where new datasets become available there may be scope for the agreement of common roles between at least two parties.

The use of role delegation may present a useful alternative for role mapping as a model of trust for NDG security of which PERMIS [10] DIS (Delegation Issuing Service) [3] is a practical implementation. Role delegation for NDG could work as follows, an SoA (Source of Authority) - in terms of NDG a data provider - adds trusted administrators of other trusted organizations to their Attribute Assignment Policy (AAP) so that the administrator may allocate roles to selected users of their organisation. This would eliminate the need for role mapping since users from the trusted organization are delegated the role needed to access the given resource. A discrete AAP would be needed over and above NDG's nearest analogue, the role mapping file held by each NDG Attribute Authority. Role delegations need to be validated.

One area where the use of role mapping could be developed would be to enable multiple levels of mapping. Currently, this is restricted to a single level. However, it is clear now that there are use-cases, particularly for international collaborations, where a second level of mapping from "federation attribute authorities" would be helpful. A future version of NDG security would allow one further mapping from specially designated federation servers.

6. Plans for the Future

The NDG team has no ambition to sustain a complete security package into the indefinite future, particularly given the range of activity in this area (e.g.[11]). At the same time, it is necessary to deploy NDG-security now, with as many applications as possible. This section addresses these two conflicting requirements.

All security paradigms consist of effectively three stages: secure credential acquisition followed by secure production of the credentials

[7] In some cases the gatekeeper is a standalone process, in others it is simply a module within the application.

in response to, or in anticipation of, a challenge, concluding with the handling of those credentials in the challenge. Hence plans for incremental change are based around handling these independently as well as addressing what it means to be "secure" while doing so.

There are two key protocols for credential establishment garnering widespread acceptance: OpenID and Shibboleth. It should not be too difficult for NDG-security to produce modified Login Servers that can support either protocol by verifying identity and then generating one-time certificates (perhaps based on Version 3 or later of MyProxy which now supports the ability to generate certificates on demand). However, before that is undertaken, there will need to be: (1) acceptance by attribute authorities in terms of supporting roles for individuals not actually registered at known data providers, and (2) gatekeepers which provide resources for such roles.

The most likely use cases for this class of acceptance is in those cases where security is being implemented to limiting access because of resource constraints, or simply to log access. It is less likely in the short term that either will be acceptable where the security constraints are based on imposing licensing or scientific constraints, where detailed data dependent roles will need to have been matched to individual users at some stage by a human.

There are a number of key applications that will need to be modified to utilise NDG-security before there is general acceptance of the benefits. Modifying such applications will depend on three steps: (1) establishing a security context either before applications are invoked, or early within them, and (2) ensuring that the security context is available subsequently within those applications, and (3) passing that context with network calls.

As well as enabling NDG and DEWS versions of the OGC web services to be NDG-security compliant, two other key applications that will be addressed in the near future are Thredds[8] and OpenDAP[9]. Both are in common use in the Met-Ocean community, primarily because of their ease of use with NetCDF archives, and providing access control to those tools would open up a large user community.

Finally, once one entertains handling digitally signed tokens as a security mechanism for regular HTTP GET transactions, the opportunity to completely remove the dependency on SOAP becomes possible. Methods of doing this include using the HTTPsec[10] emerging standard, or given that signed tokens are all that is really needed if encryption is not important, one could use simple (agreed) modifications of the HTTP headers (as is done by Google[11]). Given the paucity of existing HTTPsec implementations deploying the latter would be a pragmatic first step. That, along with Shibboleth, would result in a much simpler easy to maintain security system with only the attribute authorities and session managers as unique components.

[8] www.unidata.ucar.edu/projects/THREDDS

[9] www.opendap.org

7. References

[1] Lawrence, BN, Ray Cramer, Marta Gutierrez, Kerstin Kleese van Dam, Siva Kondapalli, Susan Latham, Roy Lowry, Kevin O'Neill, Andrew Woolf, 2004: The NERC Data Grid: "Googling" Secure Data. UK e-Science Programme All Hands Meeting (AHM2004), Nottingham, UK, 31 Aug - 03 Sept 2004

[2] SAML (Security Assertion Markup Language) version 2.0, March 2005, http://www.oasis-open.org/specs/index.php#samlv2.0

[3] Sinnott, RO,DW Chadwick, J Koetsier, O Otenko, J Watt, TA Nguyen, 2006: Supporting Decentralized, Security focussed Dynamic Virtual Organizations across the Grid. 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam, Dec 2006.

[4] Internet2 Shibboleth technology, http://shibboleth.internet2.edu;

[5] OpenID, openid.net

[6] MyProxy, grid.ncsa.uiuc.edu/myproxy

[7] Tuecke, S, V Welch, D Engert, L Pearlman, M Thompson, Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, RFC3820, June 2004.

[8] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[9] Web Coverage Service (WCS) www.opengeospatial.org/standards/wcs

[10] Chadwick, DW, A. Otenko, 2003: The PERMIS X.509 role based privilege management infrastructure. Future Generation Computer Systems, 19(2):277-289, Feb 2003

[11] Sinnott, RO, Grid Security: Middleware, Practices and Outlook, prepared for The JISC for Support for Research, Nov 2005.

[10] http://www.httpsec.org/

[11] http://code.google.com/apis/accounts/AuthForWebApps.html#signingrequests