DyVOSE Project: Experiences in Applying Privilege Management Infrastructures

J. Watt, J. Koetsier, R.O. Sinnott, A.J. Stell

National e-Science Centre, University of Glasgow
jwatt@nesc.gla.ac.uk

Abstract

Privilege Management Infrastructures (PMI) are emerging as a necessary alternative to authorization through Access Control Lists (ACL) as the need for finer grained security on the Grid increases in numerous domains. The 2-year JISC funded DyVOSE Project has investigated applying PMIs within an e-Science education context. This has involved establishing a Grid Computing module as part of Glasgow University's Advanced MSc degree in Computing Science. A laboratory infrastructure was built for the students realising a PMI with the PERMIS software, to protect Grid Services they created.. The first year of the course centered on building a static PMI at Glasgow. The second year extended this to allow dynamic attribute delegation between Glasgow and Edinburgh to support dynamic establishment of fine grained authorization based virtual organizations across multiple institutions. This dynamic delegation was implemented using the DIS (Delegation Issuing) Web Service supplied by the University of Kent. This paper describes the experiences and lessons learned from setting up and applying the advanced Grid authorization infrastructure within the Grid Computing course, focusing primarily on the second year and the dynamic virtual organisation setup between Glasgow and Edinburgh.

1. Project Background

The DyVOSE Project (Dynamic Virtual Organisations in e-Science Education) is a JISC funded two-year project investigating the establishment of a Privilege Management Infrastructure (PMI) that supports dynamic delegation of authority in the context of a Grid Computing Advanced MSc. module at the University of Glasgow. Specifically the project is investigating the application of the PERMIS software in creating an attribute management infrastructure that allows institutions to establish trust relationships that will assert and enforce the privileges presented by attributes issued by external institutions.

In the first year of the project a static PMI was implemented using the PERMIS authorization function. This allows two teams of students to author their own GT3.3 services and restrict access to certain methods provided the student held the appropriate 'team' attribute. In this case, all privileges were issued by Glasgow so no cross-organisational infrastructure was necessary [1].

In the second year the students created a GT3.3 service which ran a BLAST [2] query against a set of data retrieved from a data store hosted at Edinburgh University. Students were again split into two teams, one running a query against nucleotide data and one against protein data. PERMIS was used to secure the services at both sides, denying access to students in the protein team who attempted to extract and match nucleotide data and vice versa. In this scenario, inter-institution interaction was required, so user attributes needed to be recognized at both institutions. This may be implemented statically in the same way as the first year assignment by completely sharing user information between sites, but this is highly undesirable if we wish to deploy this kind of setup using existing campus directories. A more scalable and realistic Grid model is where local sites maintain information on their *own users* and define their own local security policies restricting access to local resources by both local users and trusted remote users/sites. The Delegation Issuing Service (DIS) aims to provide a safe, intuitive environment in which institutions may establish chains of trust without surrendering sensitive local user information. The fundamental benefit of the DIS with regard to Grids is to support fine grained authorization infrastructures whereby attributes needed for a given virtual organization can be dynamically created and recognized by remote "trusted" sources of authority. Through this model, virtual

organizations can be created in principle "on-the-fly" without detailed agreements.

2. PMI Technologies

A number of authorisation control mechanisms exist that can be applied to the Grid, examples of these include CAS [3], Akenti [4] and VOMS [5]. Each offer their own advantages and disadvantages [6]. PERMIS (Privilege and Role Management Infrastructure Standards Validation) [7] is a Role Based Access Control System [8] which uses X509 Attribute Certificates (ACs) [9] to issue privileges to users on a system. VOMS provides ACs to its users, but attributes are still handled from a central server. PERMIS is completely decentralised and access control decisions are made locally at the resource side. These access control decisions are typically made through attribute certificates (ACs) signed by a party trusted by the local resource provider. This might be the local source of authority (SoA), however a more scalable model is to delegate this responsibility in a strictly controlled manner to trusted personnel involved in the virtual organization (VO).

In a Grid context it is unrealistic to expect all information to be maintained centrally. VOs may well have many users and resources across multiple sites, and these users come and go throughout the course of the VO collaboration. Knowing for example that a given user is at Glasgow University is best answered by the Glasgow University authentication processes. However, whilst knowledge of a given users status may well be best answered by that users home authentication infrastructure, the roles and responsibilities needed to access remote resources specific to that VO may best be delegated to trusted personnel associated with that VO – it is this capability that the DIS service is to support. To achieve this requires that an authorization infrastructure exists that can firstly define appropriate policy enforcement (PEP) and policy decision points (PDP), i.e. define and enforce the rules needed to grant/deny access requests based upon ACs .

PERMIS offers a generic API which can be applied to any resource, so our investigations could also be applied to non-Grid regimes.

The PERMIS decision function can be a standalone Java API, or it can be deployed in the same container as the Grid Service it is intended to protect. The GGF SAML AuthZ API [10][11] provides a method for Globus to bypass the generic GSI [12] access control and allow external services to make authorisation decisions. Once deployed, the PERMIS service requires an XML policy which describes in complete detail the targets, actions and roles which apply at the resource (or at the institution). This policy may be written using the Policy Editor GUI supplied with the PERMIS software, or may be edited by hand. Another important GUI supplied with PERMIS is the Privilege Allocator (or the slightly more user friendly Attribute Certificate Manager (ACM)). This is responsible for allocating roles and signing ACs for users. This tool can also be used to browse LDAP directories for ACs and can be useful in confirming that ACs have been loaded correctly.

In order to implement a dynamic PMI, extensions to this ACM tool need to be made. As it stands, the ACM can issue any certificate it wishes, irrespective of its validity within the PMI. Ideally a method of enforcing the infrastructure described in the XML Policy to allow an administrator to only be allowed to issue valid ACs is needed. To keep user information at the home site, it would be necessary to have a mechanism that would allow a remote administrator to issue ACs only with roles relevant at their institution to the home site LDAP. There are two gains to this, the first being that the remote admin can only operate within a very restricted attribute set, which would exclude any possibility of them issuing home site roles. The second gain is that, as requested, all important user data is still held at the home institution.

The Delegation Issuing Service (DIS) is intended to provide this functionality. The DyVOSE project is the first project to investigate this technology to any great extent, with the goal of providing a user and admin guide to the installation and operation of the DIS service. In the next section we describe some of the technicalities of setting up and using the DIS and then we outline how we have applied this technology for teaching purposes.

3. The Delegation Issuing Service

In the current implementation, the DIS software is a web service consisting of a Java library based on the Tomcat and AXIS SOAP servers. The web service is accessed by a DIS client written in PHP running on an Apache server which acts as a proxy between the DIS service and the user. This client invokes the Java component through SOAP calls, and is presented to the user in a web browser after mandatory authentication to Apache using their own username and password. These components may be hosted on separate

machines, although for the DyVOSE investigations, they were situated on the same computer.

The DIS service assumes that there is a Tomcat application server of recommended version 4.x installed on the server side, along with an LDAP server for attribute storage and Apache authentication. A Java Runtime Environment of at least version 1.4 is required. On the client side, a functional Apache web server loaded with the SSL, PHP and LDAP modules is required. The Apache and Tomcat servers were hosted on the same server with no incompatibility issues encountered. The resource OS was chosen to be Fedora Core 4 as this distribution contained all the Apache functionality and extra modules as standard RPMs and were loaded (and to some extent, configured) automatically. Edinburgh has successfully (in terms of providing default functionality) migrated the DIS service to Fedora Core 5. In addition, the LDAP backend (Berkeley DB) was of a version advanced enough to allow the most recent version of OpenLDAP to be installed on the machine. For the purposes of the Grid Course assignment at Glasgow, this was abandoned in favour of a slightly older version of LDAP which was compatible with the GT3.3 PERMIS Authz service deployed on a separate machine, although the Edinburgh DIS was successfully integrated with the newest version of LDAP.

The DIS software itself ships as two gzip files containing the server and client side tools separately. These files include the necessary Java libraries, Web Service Deployment Descriptor file, LDAP core schema, and configuration files. A sample LDIF file containing the required DIS users and their certificates is provided for loading the LDAP server to test the installation. Due to the complexity of the surrounding PKI, this file is essential for installation as it is HIGHLY unlikely that a DIS-friendly certificate infrastructure could be deployed prior to confirming the success of the DIS install. One drawback with this is that using this file forces the DIS service to handle users with a fixed Distinguished Name (DN) which makes that particular setup quite non-portable.

The implementation of the DIS requires a consistent PKI comprising a total of around 9 certificates and key pairs in order to realise the service and its proxy. In several cases, the certificates need to be loaded in three different formats (PEM, DER and p12) [13] in order to talk to the various components of the DIS service, and the underlying PERMIS server that the DIS creates ACs for. These certificates are created from the command line using *openssl* after creation of a site specific configuration file which handles certificate extensions and populates the DN with a structure corresponding to the users present in LDAP. The two key users in the DIS infrastructure are the Source of Authority (SoA) and the DIS user. These are the only two users who require pre-loaded ACs as everyone else in the LDAP server can be allocated ACs by the DIS. The AC is stored as an attribute labelled *attributeCertificateAttribute*, with an optional *;binary* extension dependent on the local LDAP schema. The AC is created using the PERMIS Attribute Certificate Manager (ACM) GUI, which can also load the certificate into LDAP. These two pre-loaded ACs are essential to the operation of the PMI, and for the SOA and DIS user, they contain an attribute *'permisRole'* whose entries are a list of all assignable roles within the infrastructure. In the case of the DIS, the attribute list is an explicit statement of every role that the DIS can assign and delegate. In addition to the attribute list, the SOA requires another attribute called *'XML policy'* which contains the XML file representing the site policy. This policy states the hierarchical relationships between roles, which targets and actions these roles apply to, the scope references, and which SOAs are to be trusted within the VO.

The SoA is the root of trust for the PKI, and signs every certificate (and through the DIS, every Attribute Certificate) within the infrastructure. A PEM format root certificate was created using *openssl*, along with its corresponding encrypted private key. This certificate is required to be present in the Apache SSL configuration and is used to create user certificates compatible with the Globus Toolkit, allowing Grid users to interact with the PMI and be assigned meaningful privileges. The root certificate is required to be loaded into the SOA node of the LDAP server, in this case the PEM certificate needs to be converted to the DER format using openssl. In LDAP, this certificate is loaded under the *"userCertificate"* attribute, again with an optional *";binary"* suffix depending on the version of LDAP. In addition, this DER format file is required by the DIS server for validation, and also needs to be loaded into the Tomcat server keystore and the Java JRE security CA keystore. Finally, the root certificate and its private key need to be converted to a PKCS p12 format which is used by the Attribute Certificate Manager (ACM) to sign the SoA ACs.

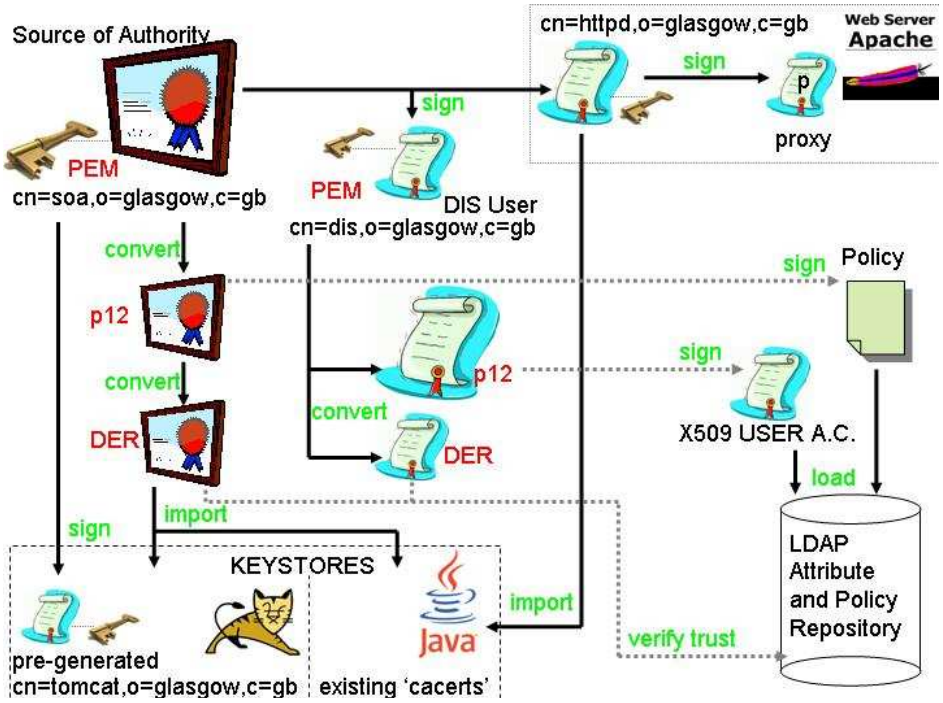

Figure 1: A schematic of the underlying PKI within the DIS service.

The DIS user is the other key entity within the PMI, and it is this user who is responsible for signing ACs and assigning and revoking privileges. In the same way as the SoA, the DIS user is assigned a certificate matching its LDAP DN, and this certificate is converted to DER format and loaded into LDAP using the '*userCertificate*' attribute as above. This is to complete the validation chain for the PERMIS server when it attempts to verify the credentials of a presented user certificate. The DIS certificate is converted to PKCS12 format and loaded into the Tomcat configuration directory.

Two keystores need to be maintained within the DIS service. The generic Java JRE security CA trust keystore needs to be loaded with the root certificate (in DER format). A second keystore for the Tomcat server needs to be created using the Java keytool. A private key and certificate for the tomcat server is created and loaded into the keystore, along with the root certificate (DER) again to verify the certificate. A schematic of the Public Key Infrastructure required to realise a basic DIS service is shown in Figure 1.

The DIS service is operated by logging in to the DIS proxy which retrieves its authentication information from LDAP. Any ACs issued by this service are always signed by the DIS user, as it is his certificate (p12 file) which is in the signing certificate repository. However, information on the user that logs in will determine which roles can be assigned or delegated. The SoA has the ability to delegate and grant all privileges within the PMI, but delegated users, including the DIS, will only be able to delegate those roles that they possess and are authorised to delegate. In the same way, the DIS web service will only sign certificates and issue roles if the user logging into the DIS possess the relevant credentials and is authorised to delegate those roles..

To understand how this infrastructure facilitates dynamic establishment of security focused virtual organisations, we consider the Grid programming assignment at Glasgow University.

4. Implementation Scenario

As part of the Grid Course programming assignment, students at Glasgow authored a GT3.3 Grid Service which used BLAST to do similarity searching on a set of either protein or nucleotide data retrieved from a GT3.3 service based at the National e-Science Centre at the University of Edinburgh. These services were PERMIS protected to only grant access to students in the same group (Protein Team P or Nucleotide Team N) provided they presented an AC with the correct role. Initially, these ACs were granted in advance and stored at both

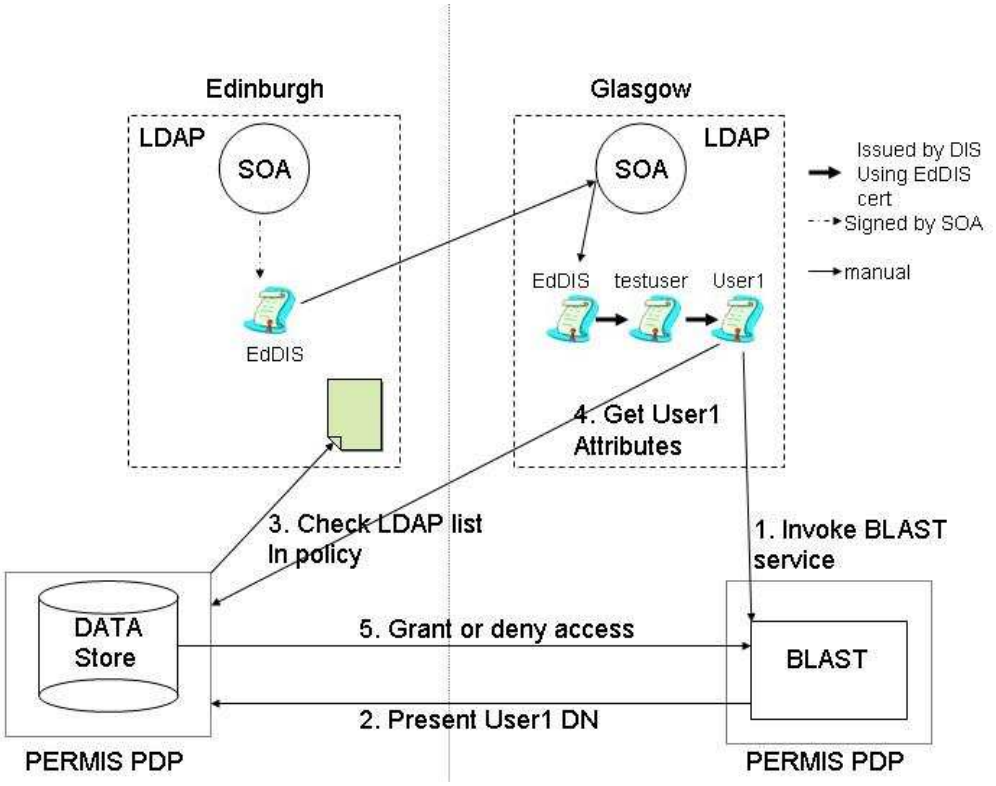

Figure 2: Diagram of the interactions required for Edinburgh to issue an AC granting access to its resources.

locations for testing the service functionality. This is undesirable as in general, information on students should only really be present at the student's home institution. The benefit of this approach is that the implementation of this static PMI is well understood and easily maintained through expertise gained in the previous year's work.

To extend this static PMI to a model supporting dynamic delegation, a DIS service was created at both sites. Using the Glasgow DIS, the Edinburgh SOA could login and grant Glasgow users the privilege to access the Edinburgh data store. This way Glasgow retains all its student's details, yet through privilege delegation, an Edinburgh user can grant these users a role recognised by Edinburgh provided they have been granted this privilege by the Glasgow DIS.

A number of different approaches were considered based on the current version of the PERMIS software. One method which was attempted was the use of LDAP referral, which would allow a single LDAP to be specified by the Edinburgh PERMIS Policy Decision Point (PDP) for it to retrieve its user attributes from. With referral set up, a branch of the Edinburgh LDAP server would point to the Glasgow LDAP as if it were part of its own tree. This approach ran into several problems, the main one being that when the PERMIS PDP searched the LDAP tree and came across a referral, the LDAP server bounced the details back to the PDP for it to do the search itself on the remote LDAP. Since no functionality exists for this the PDP crashed each time it encountered this referral. Attempts to make this referral transparent to the PDP, i.e. getting the LDAP server to do the retrieval and presenting remote attributes as if they came from the local LDAP were not successful. The PERMIS team assured us that the PDP LDAP server parameter could take several values, however despite numerous attempts this was never made to work. A solution was found in which multiple LDAP servers could be listed within the site policy itself under a "Repository Policy" subject tag. This method meant that referral was not necessary nor did it compromise local security since the entry was merely a location in which to look for attributes, and not a statement of trust on the part of the local SoA.

Two aspects of dynamic delegation which at the time of writing were not implemented in the PERMIS software were those of role mapping and authority recognition. Role mapping allows separate sites with their own security policies to

state which roles that apply at one site can match the roles at another site. Typically, an institution will define "External" roles that have lower privilege than the local ones and these roles are typically used as equivalences. Since this functionality was not present on implementation, it was forced upon the PMI by an agreement which stated that the external roles at both institutions would be given exactly the same name. Now the only difference between an external Glasgow role and an external Edinburgh role is that of which SoA (or in this case, DIS) actually signed the user's AC.

The second function which was not available yet was that of recognition of authority, or how the VO formed between the two sites would recognise ACs signed by the other site. An easy solution to this is to add the external SoA to the "SoA Policy" tag within the XML policy. This way, any ACs extracted which have been signed by the remote host site can be verified. This method, although easy, means that a given site explicitly trusts all of the actions of the remote SoA. Without a DIS service protecting the assignment of ACs, the remote SoA could in principle assign any role they are aware about from the home institution to any of its users. The DIS service, since it only ever issues valid ACs within the constraints of its own site policy, can enforce more stringent rules on what the remote SoA can allocate. However, we suggest a different approach which has been implemented in our dynamic delegation scenario.

Instead of trusting an external SoA to establish a chain of trust, we created a *EdDIS* user at the remote host who has been allocated a DIS certificate containing all the roles they may delegate, but which has been signed by the home institution. Therefore when a Glasgow user presents an AC which has been signed by this *EdDIS* user (within the Glasgow DIS) this AC will already be trusted by Edinburgh.

To understand this we provide an example of granting access to the Edinburgh data to a Glasgow user. This sequence is shown pictorially in Figure 2, with the PERMIS decisions on the Glasgow side being omitted as this is a purely static PMI function.

The Edinburgh SoA creates an "*EdDIS*" signing key pair which is signed by itself. This certificate is handed to the Glasgow SoA (via a secure channel) and the administrators on the Glasgow side mount this certificate in their LDAP directory. The Edinburgh SoA also creates an AC, issuing two external roles and the ability to delegate those roles, to this user, "*GlaStudentTeamN*" and "*GlaStudentTeamP*. Now an extra user *EdDIS* appears in the Glasgow DIS. To demonstrate delegation, a new user called "*testuser*" was created at Glasgow, who was issued with an AC signed by *EdDIS* which allowed the user to delegate the external roles to other Glasgow users (in this case "*User1*"). The "testuser" can then log into the Glasgow DIS and create an AC for User1 containing the role "*GlaStudentTeamN*".

User1 calls the Edinburgh Grid Service through their own Glasgow BLAST service. GSI passes User1's DN to the Edinburgh PERMIS PDP. The PDP reads the Edinburgh policy, and locates the LDAP server to extract the User1 attributes (contained in the Repository Policy list). The remote LDAP is queried, and the *User1* AC is extracted. The PDP checks the signature on the AC and verifies that it has been signed by the *EdDIS* user, who although existing at Glasgow, is signed by the Edinburgh SoA. Since the chain of trust can be verified back to the Edinburgh SOA, who is the trusted SoA in their policy, the PDP establishes this is a valid AC. Then the PDP makes the decision based on the user attribute presented whether to release the Protein or Nucleotide data to the Glasgow Grid Service. Any ACs, that are part of the Glasgow PMI, issued by the Glasgow SOA, will simply be ignored by the Edinburgh PDP, as they are not signed by any party that the Edinburgh SOA trusts. This allows two PMIs to co-exist in one LDAP tree.

The DIS can assign and revoke user ACs as many times as it wishes, without affecting any other user certificates in its infrastructure. Also, any users who have delegated their roles to other people will find that the roles they delegated will still be valid even if they cease to be members of that PMI. A screenshot of the DIS service window is shown in Figure 3.

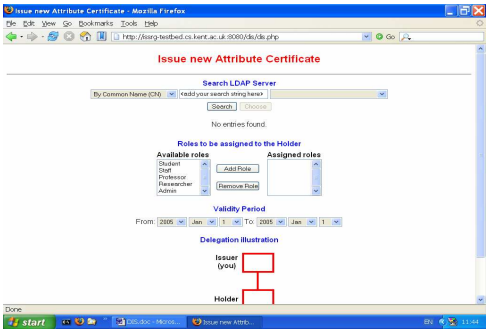

Figure 3: The DIS service Web Interface

5. Experiences and Conclusions

The DIS software shows great promise as a tool to enable dynamic VO establishment. We have successfully demonstrated a VO which allows a SoA at a remote site to securely assign and revoke privileges to home users, without user information being factored out to external databases. The adherence of the DIS service to the local policy means that only valid ACs can be issued, and the scenario described above allows the establishment of distributed trust without surrendering local security. Once installed the service is intuitive, with a GUI interface that allows AC issuing in a few seconds.

The Grid Computing module is now completed. Of the 11 students that took this module this year, all but one managed to access and retrieve data from the PERMIS protected service in Edinburgh, thus providing that the infrastructure works.

The work has not been without issues however. The lack of availability of source code due to commercial concerns makes the tracking of errors and diagnosis of problems very problematic. This extended the development time of this project by an unacceptable amount due to our reliance on the tireless help of the PERMIS development team, in particular Sassa Otenko whom we are grateful to for his efforts. The underlying PKI to establish the DIS service is over-complicated, most certificates are required to be duplicated and converted many times through the system. This is probably due to the Web Proxy based approach of the GUI, which demands many certificates and keystore entries to be maintained. Some of our scenario definitions have had to change on several occasions due to undocumented features within the DIS and PERMIS.

In the absence of source code, some heavier documentation would be desirable, in particular with regard to setting up the PKI. A simpleCA approach that could generate the appropriate keys, in the appropriate formats according to the domain structure of your institution would be invaluable. Once running, the software is easy to use and robust, but the implementation time required at this stage of the software development may be outside the remit of any developers who are new to this technology.

Nevertheless the proof of concept that dynamic delegation of authority works has been a major output of the project. We believe that this federated model of policy definition and management that suit the needs of a multitude of VOs has numerous potential application areas. One key area of focus is to support and extend the largely static nature of ACs used for example in Shibboleth identity and service provider interactions. Shibboleth access to resources has up to now been based largely upon an agreed set of attributes and their values based for example around the eduPerson object class (www.eduperson.org). This model is not conducive to the more dynamic nature of short lived Grid based VOs which come together for a given time to solve a particular problem. In this case, dynamic creation and recognition of attributes based on a limited trust model is more apposite. As such, we plan to explore this technology in a range of other e-Science projects at the National e-Science Centre in Glasgow.

5.1 Acknowledgements

The DyVOSE project was funded by a grant from the Joint Information Systems Committee (JISC) as part of the Core Middleware Technology Development Programme. The authors would like to thank the programme manager Nicole Harris and collaborators in the project. In particular special thanks are given to Professor David Chadwick and especially Dr Sassa Otenko for help in exploring the DIS and PERMIS technologies.

6. References

[1] R.O.Sinnott, A.J.Stell, J.Watt, “Experiences in Teaching Grid Computing to Advanced Level Students” Proceedings of CLAG+GridEdu Conference, May 2005, Cardiff, Wales

[2] BLAST (Basic Local Alignment Search Tool), http://www.ncbi.nih.gov/Education/BLASTinfo/information3.html

[3] L. Pearlman, et al., “A Community Authorization Service for Group Collaboration” in Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002

[4] Johnston, W., Mudumbai, S., Thompson, M., “Authorization and Attribute Certificates for Widely Distributed Access Control”, IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, CA, June 1998, p340-345 (http://www-itg.lbl.gov/security/Akenti)

[5] VOMS Architecture, European Datagrid Authorization Working Group, 5th September 2002

[6] A.J.Stell, “Grid Security: An Evaluation of Authorisation Infrastructures for Grid Computing” MSc Dissertation, University of Glasgow 2004

[7] Privilege and Role Management Infrastructure Standards Validation project (www.permis.org)
[8] D.W. Chadwick, O. Otenko, "The PERMIS X509 Role Based Privilege Management Infrastructure", Future Generation Computer Systems, 936 (2002) 1-13, December 2002, Elsevier Science BV
[9] D.W.Chadwick, O. Otenko, E.Ball, "Role Based Access Control with X.509 Attribute Certificates", IEEE Internet Computing, Mar-April 2003, pp. 62-69
[10] V. Welch, F Siebenlist, D.Chadwick, S. Meder, L. Pearlman, "Use of SAML for OGSA Authorization", June 2004, https://forge.gridforum.org/projects/ogsa-authz
[11] OASIS, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) v1.1, 2 September 2003, http://www.oasis-open.org/committees/security
[12] Globus Security Infrastructure (GSI) http://www.globus.org/security
[13] OpenSSL: http://www.flatmtn.com/computer/Linux-SSLCertificates.html

Author/s:
Watt, J.;Koetsier, J.;Sinnott, R. O.;Stell, A. J.

Title:
DyVOSE project: experiences in applying privilege management infrastructures

Date:
2006

Citation:
Watt, J., Koetsier, J., Sinnott, R. O., & Stell, A. J. (2006). DyVOSE project: experiences in applying privilege management infrastructures. In Proceedings of the UK e-Science All Hands Meeting, Nottingham, UK.

Publication Status:
Published

Persistent Link:
http://hdl.handle.net/11343/28811