Authorisation in Grid computing

David Chadwick*

University of Salford, UK

Abstract
This paper briefly surveys how authorisation in Grid computing has evolved during the last few years, and presents the latest developments in which Grid applications can utilise a policy controlled authorisation infrastructure to make decisions about which users are allowed to perform which actions on which Grid resources. The paper describes the Global Grid Forum SAML interface for connecting policy based authorisation infrastructures to Grid applications, and then describes the PERMIS authorisation infrastructure which has implemented this interface. The paper concludes with suggestions about how this work will evolve in the future.

Introduction
Grid computing allows resources, including large scale expensive ones such as genetic databases, to be shared between members of virtual organisations (VOs). However, if an organisation is to allow its resources to be shared amongst its VO partners, it needs to be able to determine who is authorised to access these resources in which ways, and who is not. Ideally each resource owner should be able to set the policy determining the rules for who is authorised to do what, and then leave the authorisation infrastructure to enforce this policy. The resource owner should not be expected to individually identify and name each VO user who is to access his/her resource, as this soon becomes unwieldy and costly to manage. The resource owner should be able to delegate to other partners in the VO the ability to identify and nominate the users from their respective domain who are to be allowed to use his/her resource, leaving him/her to simply determine what type of access to grant to the different categories of user. It is only very recently that we have been able to achieve this, as will be described here.

The rest of this paper is structured as follows. Next section provides a brief history of Grid authorisation. Then the current Global Grid Forum draft authorisation interface is described. Further the PERMIS policy based authorisation infrastructure that is compatible with the GGF interface is described. Finally last section concludes and looks at possible future work in this area.

* Present address: Computing Laboratory, University of Kent, Kent, UK.
E-mail address: d.w.chadwick@kent.ac.uk

1363-4127/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.istr.2004.11.004
A brief history of Grid authorisation

One of the earliest attempts at providing authorisation in VOs was in the form of the Globus Toolkit Gridmap files (Sotomayor). This file simply holds a list of the authenticated distinguished names of the Grid users and the equivalent local user account names that they are to be mapped into. Access control to a resource is then left up to the local operating system and application access control mechanisms. As can be seen, this neither allows the local resource administrator to set a policy for who is allowed to do what, nor does it minimise his/her workload. On the contrary it maximises the work of the resource administrator since (s)he must first pre-configure the Grid application with the names of every VO user who is to be allowed to access the Grid resource, and then (s)he must set the access controls on the local operating system and/or application to ensure that the local user names are restricted in what they are allowed to do with the resource. The system is neither scalable nor flexible, and there is no way to distribute the administrative task throughout the VO. Consequently several ways were developed to improve upon this.

The Community Authorisation Service (CAS) (Pearlman et al., 2002) was the next attempt by the Globus team to improve upon the manageability of user authorisation. CAS allows a resource owner to grant access to a portion of his/her resource to a VO (or community — hence the name CAS), and then let the community determine who can use this allocation. The resource owner thus partially delegates the allocation of authorisation rights to the community. This is achieved by having a CAS server, which acts as a trusted intermediary between VO users and resources. Users first contact the CAS asking for permission to use a Grid resource. The CAS consults its policy (which specifies who has permission to do what) and if granted, returns a digitally signed capability to the user containing details about what the user is allowed to do (as an opaque string). The user then contacts the resource and presents this capability. The resource checks that the capability is signed by a known and trusted CAS and if so maps the CAS’s distinguished name into a local user account name via the Gridmap file. Consequently the Gridmap file now contains other privileges or group membership details, in which case software would be needed by the resource to interpret this information and grant appropriate rights. The software they developed for this is called the Local Centre Authorisation Service (LCAS) (Steenbakkers, 2003). LCAS makes its authorisation decision based upon the user’s certificate and the job specification, which is written in job description language (JDL) format.

In its current re-incarnation, the VOMS server now produces short-lived X.509 attribute certificates (ACs) (ISO 9594-8/ITU Rec. X.509, 2001) for the user to push to the resource. This design is similar in concept to the CAS, but differs in message format and syntax. In VOMS the name of the user is presented to the resource instead of the name of the CAS server, and user attributes are presented instead of opaque policy statements. The message construct is signed by the VOMS server instead of the CAS server, and is a standard X.509 AC instead of a proprietary capability. However, what neither VOMS nor CAS nor LCAS provides is the ability for the resource administrator to set the
policy for access to his/her resource and then let
the authorisation infrastructure enforce this policy
on his/her behalf. This is what systems like Akenti
(Johnston et al., 1998), PERMIS (Chadwick et al.,
2003), and Keynote (Blaze et al., 1999) provide.

The Globus team realised that ultimately this is
what is needed for Grid authorisation, and so,
within the remit of the Global Grid Forum (GGF),
set about defining a standard interface between
a Grid application that wants to know if a user is
authorised to perform a certain action, and an
authorisation infrastructure that is able to answer
such questions.

124 An authorisation interface for the Grid

The ISO Access Control Framework standard (ITU-T
Rec X.812, 1995) recognised nearly a decade ago
that access control can be split into two compo-
nents: an application dependent enforcement
function (AEF) and an application independent
decision function (ADF) — termed, respectively,
the policy enforcement point (PEP) and policy
decision point (PDP) in various IETF documents.
The PDP (or ADF) can be controlled by a policy (or
set of rules), and providing the policy is general
enough, the PDP is able to make decisions for any
type of application. All the ADF needs in addition
to the policy is details about the user (initiator),
the resource being protected (target), the access
request and environmental (or contextual) infor-
mation such as the time of day (see Fig. 1). In
addition, if the ADF/PDP retains information about
previous requested actions it can make subsequent
decision based on this, for example, ATM machines
that will only allow you to withdraw a maximum
amount of money each day and will refuse re-
quests once this limit has been reached. The Open

Group defined an Application Programmable In-
terface between the PEP and PDP and called it the
AZN API (The Open Group, 2000). However, this is
not general purpose enough, being constrained
to C programs only.

The Grid is increasingly moving towards web
services as the means of connecting its various
components together. Globus Toolkit is moving this
way too, and release 4 scheduled for the end of
2004, is expected to be web services compliant. In
2003 it was therefore opportune to specify a web
services interface that could be used to connect
a PDP to a Grid based PEP such as Globus Toolkit.
Fortunately the groundwork for this had already
been done by the Organization for the Advance-
ment of Structured Information Standards (OASIS),
who had issued the Security Assertion Markup
Language (SAML) specification in 2002 (OASIS,
2002a). SAML is a general purpose language that
allows different types of security assertions about
principals to be passed between clients and serv-
er, encoded as XML messages. The language is
infinitely extensible and allows any type of asser-
tion to be defined, although three standard types
of assertions about principals were specified in
SAML v1.0: authentication assertions, attribute
assertions and authorisation decision assertions.

It is the latter type that are passed between the
PEP and PDP. SAML therefore provides a solid base
from which to specify the PDP—PEP interface for
Grid applications. The SAML specification also
defines a request—response protocol in SOAP over
HTTP for carrying the SAML assertions. SAML is thus
fully web services compliant.

Whilst SAML provides a solid basis for specifying
the PDP—PEP interface, it does not define every-
thing that is needed for Grid applications. A SAML
profile is thus needed to rectify the deficiencies,
and this has been specified by the Global Grid
Forum as a draft standard specification (Von Welch
et al., 2004). Several important restrictions or
additions to SAML have been specified, including:

- The contents of the authorisation response. A
  simple Boolean (granted or denied) is sufficient
  for the PDP—PEP interface, but SAML does not
  contain such a response,1 hence the GGF
  profile defines one.
- How the PDP gains access to the user’s auth-
  1 The SAML Authorization Decision Response repeats the
  entire contents of the Authorisation Decision Request, which
  is useful if the request is sent by a party other than the PEP, for
  example, the principal. The SAML response thus details exactly
  what has been granted to the principal.

  Figure 1 Separating access control enforcement from
decision making.
modes are possible, pull and push. In pull mode
the PDP fetches the credentials, in push mode
the PEP provides them. If the PDP is to fetch
the credentials, how does it know where to get
them from? The PDP could either be pre-
configured with a (probably static) list of
credential sources, or the client could tell the
server where to pull the credentials from at the
time of decision making. The latter is more
scalable, and more dynamic than the former.
Thus the GGF profile defines a Reference
Statement which points to a repository where
user credentials are located.
- Default values for all the parameters of the
authorisation decision request. These are
specified in the GGF draft as follows: if the
name of the initiator is missing it is assumed to
be anyone i.e. public access is being re-
quested; if the requested action is missing it
is assumed to be everything i.e. all the rights
that have been granted to the initiator; if the
target is missing it is assumed to be all the
resources that are protected by the PDP policy;
if the initiator’s credentials are missing then
only the default ones (if any) that have been
granted to everyone should be used. Note that
no default values for the contextual informa-
tion have been specified since in general it is
not possible to define these.
- If too little information is passed to the PDP, it
may simply deny access. Alternatively, at its
discretion, the PDP may return “Granted sub-
to” along with a set of conditions that
must be fulfilled before access is granted, e.g.,
Granted subject to the time being between 9
am and 5 pm. It is then the responsibility of the
PEP to evaluate these conditions before grant-
ing access to the initiator. If the PEP is unable
or unwilling to evaluate these conditions, it
always has the option of issuing a new decision
request and sending more information to
the PDP (such as the missing contextual
information).

Once this interface had been defined by the
GGF, it then needed to be implemented and tested
in one or more Grid applications to ensure that it
meets the needs of the Grid community. Globus
Toolkit v3.3, released in April 2004, has imple-
mented this SAML interface, as has the PERMIS
authorization infrastructure, described in the next
section. The BRIDGES E-Science project currently
running at Glasgow University is the first Grid
application to pilot the combined GT3.3/PERMIS
infrastructure and the results are expected to be
published the last quarter of 2004.

The PERMIS authorisation infrastructure

PERMIS is software developed under the EC PERMIS
project (www.permis.org). It is now part of the US
National Science Foundation’s Middleware Initiati-
ve (NMI) software release (www.nsf-middleware.
org). PERMIS is an attribute based access control
(ABAC) infrastructure. ABAC is a superset of role
based access controls (RBAC), in which access
control decisions are made based upon any attributes
held by the user, and not just upon their
organisational roles (as in conventional RBAC). In
PERMIS, user attributes are held in X.509 standard
attribute certificates (ACs) (ISO 9594-8/ITU Rec.
X.509, 2001). An attribute certificate is a data
structure that binds details about the holder to the
attributes that are assigned to them, digitally
signed by the issuing attribute authority. The AC
is therefore tamper-proof, and its validity can be
checked by validating its digital signature, and
checking that it has not been previously revoked in
the current revocation list.

In PERMIS, managers throughout a VO can act as
attribute authorities and assign attributes (in the
form of X.509 ACs) to their staff — they do not
need any prior permission to do this. All they need
is a private signing key and a corresponding X.509
public key certificate (plus the necessary software
to create ACs). Thus the allocation of entitlements
(or ACs) is distributed throughout the entire VO
(and beyond if necessary\(^2\)). However, each re-
source owner, when setting the policy that con-
trols access to his/her Grid resource, states which
attribute authorities(s)he trusts to issue X.509
ACs, and the PERMIS PDP will then discard all ACs
presented to it that are not digitally signed by one
of these trusted authorities. Thus we have partly
accomplished one of our earlier stated goals, i.e.
that a resource owner should be able to specify
a policy for controlling access to his/her resource,
and then leave the authorisation infrastructure to
enforce it.

The PERMIS distribution contains three software
tools for creating X.509 ACs. A user friendly graph-
ical Attribute Certificate Manager (see Fig. 2)
is designed to make it very easy for managers to
assign basic ACs to their staff, one by one. A more
sophisticated Privilege Allocator can create more
complex ACs, whilst a bulk loader tool is designed to
allow large numbers of users to be automatically

\(^2\) PERMIS does not restrict who can issue ACs. Thus, for
example, a professional society such as the Law Society or the
General Medical Council, could issue “lawyer” or “doctor” ACs
to their members, and access control decisions could sub-
sequently be based upon them.
allocated ACs. The bulk loader works by searching an LDAP directory for users who have specific attributes, then creating a specific AC for each of them and writing these back to their respective LDAP entries.

PERMIS provides a PDP that reads in the policy set by the resource owner at initialisation time. It then makes access control decisions for each authorisation decision request provided by the PEP, based on this policy. The interface between the PDP and the PEP is implemented as a Java API (for applications that want to combine the PDP and PEP as one program), and as SAML requests over SOAP and HTTP (for stand alone PDP servers, as used by GT3.3).

The user’s X.509 ACs can be pushed to the PERMIS PDP by the PEP, either as complete X.509 ACs, or as SAML Reference Statements. The PDP can also pull X.509 ACs from a set of pre-configured LDAP servers, or from the locations specified in the Reference Statements.

PERMIS policies are written in XML, according to a DTD published at www.xml.org. This DTD predates the XACML specification (OASIS, 2000b), and for the most part is a subset of XACML, except that the PERMIS policy supports delegation of authority, a feature not currently supported by XACML. The PERMIS authorisation policy is a set of sub-policies, namely:

- SubjectPolicy — this specifies the valid subject domains i.e. only users from these subject domains may be authorised to access resources covered by this policy. Each domain is specified as an LDAP subtree, using Include DN and Exclude DN statements. In Grid environments each user has a unique DN contained in his public key certificate. If his DN is not within a valid subject domain, the user will be denied all access to resources in the VO covered by this policy.

- RoleHierarchyPolicy — this specifies the different roles and attributes that can be allocated to users, and the hierarchical relationships (if any) between them. A superior role inherits all the privileges of a subordinate role, as in conventional RBAC. Using role hierarchies can simplify Role Assignment Policies.

- SOAPolicy — this specifies which attribute authorities (or Sources Of Authority) are trusted to allocate roles and attributes to users. This is the way that a resource owner specifies who within (and without) the VO is to be trusted to issue ACs.
RoleAssignmentPolicy — this specifies which roles and attributes may be allocated to which subjects by which SOAs, whether delegation of authority may take place or not, and how long the issued ACs are considered valid by this policy. This sub-policy effectively states who is trusted to allocate which roles to whom, and is central to the distributed management of trust. It can stop a trusted manager in one organisation issuing attributes or roles to staff in another organisation. Allowing delegation will allow a staff member to assign his/her attributes to another staff member. By restricting the validity of ACs, an issued AC may have a validity period of 2 years, but the resource owner may have a more stringent policy and be not prepared to accept ACs older than 1 year.

TargetPolicy — this specifies the target domains in which resources covered by this policy are located. Each domain is specified as either an LDAP subtree, using Include DN and Exclude DN statements, or as a URL. Resources can further be refined by specifying their type e.g. all fileservers within the o=Salford, c=GB domain.

ActionPolicy — this specifies the actions (or methods) supported by the various target resources, along with the parameters that should be passed along with each request e.g. action Open with parameter Filename.

TargetAccessPolicy — this specifies which roles and attributes are needed in order to perform which actions on which targets, and under which conditions. Note that this part of the policy, being ABAC, is not concerned with the distinguished names of the users. Conditions are specified using Boolean logic and might contain constraints such as "IF time is GT 9 am AND time is LT 5 pm OR IF Calling IP address is a subset of 125.67.x.x". All actions that are not specified in a Target Access Policy are denied.
Clearly setting the policy for controlling access to a resource is a potentially complex task, especially if the resource owner was required to write the XML to specify it. This would be beyond the capabilities of most resource owners. Consequently, a user-friendly GUI is provided to make it easy to write PERMIS policies (see Fig. 3). Once the policy is written, the resource owner digitally signs it (to prevent it from being subsequently tampered with), and the GUI stores it in the owner’s LDAP directory entry. Each policy also has a unique ID so that an owner can have several different policies for different occasions. When the PERMIS PDP is started it is told which policy ID to use, so it reads in this policy from the owner’s LDAP entry, checks the signature and validity time, and if valid, the PDP knows that it has the correct policy written by the resource owner. The resource owner can then leave the authorisation infrastructure to control access to his/her resource without having to continually manage the system.

416 Conclusions and future work

Allowing a resource owner to set the policy for who has permission to perform which actions on which resource, and distributing the assignment of rights to managers throughout a VO, is clearly something that is needed to facilitate large-scale Grid computing. Separating access control enforcement into a policy controlled application independent PDP and application dependent PEP, linked together via a web services SAML interface is something that will facilitate the development of comprehensive and sophisticated PDPs such as PERMIS and Akenti. As Grid resource owners demand more access control features, such as separation of duties, mutually exclusive roles, dynamic delegation of authority, etc., then it becomes a question of adding suitable rules to the policy base and enhancing the PDP to correctly evaluate them.

At this point in time it is not clear which XML based policy language will become the de-facto standard for PDPs. XACML (OASIS, 2000b), from OASIS has some industry support, and Grid researchers are now experimenting with it. But Microsoft is also championing XrML (www.xrml.org) as a general purpose language for expressing who has the right to do what with digital content. Clearly there is significant overlap between these two languages, but with XrML now being part of Microsoft Office 2003 and Windows Server 2003, it could very well become the de-facto standard for authorisation. The downside of XrML is that a number of its features are patented, so users will need to pay a license fee to use it.

Future work inside the GGF is likely to specify a management interface to the PDP that will allow the resource manager to dynamically update the policy that is to be used for decision making. In current systems it is a local matter how the PDP is configured with the correct policy to be used, and how the policy is changed according to the changing circumstances in the VO. Further, in the future, autonomic PDPs could well be developed and configured with meta-policies telling them which access control policy to use under which conditions. As one can see, policy based authorisation and access controls are definitely here to stay.

References

Johnston W, Mudumbai S, Thompson M. Authorization and attribute certificates for widely distributed access control.

Professor David Chadwick is the leader of the Information Systems Security Research Centre (ISSRC) at the University of
Salford. This is part of Informatics Research Institute (see http://www.iris.salford.ac.uk/) which gained a 5* rating in the December 2001 RAE. Prof Chadwick has written over 40 books, journal and conference papers and the latest of these can be downloaded from http://sec isi.salford.ac.uk/Papers. He was the editor of X.518 (1993) and is the author of several current Internet and Global Grid Forum Draft Standards including the Grid authorisation API draft standard. He has been the principal investigator in over 10 research grants from a variety of sources including JISC, EPSRC and the EC. He was the technical director of the EC PERMIS project (www.permis.org) and instrumental in its design and evolution.