

An Authorisation Interface for the GRID

D.W.Chadwick, University of Salford.

Abstract

The provision of one or more separate authorisation infrastructures, comparable to the existing Grid authentication infrastructure, is desirable, since it will allow Grid applications to plug and play different authorisation infrastructures in order to choose the best one for their needs. The first half of this paper describes the features that are needed from this interface. Whilst it is possible to standardise every conceivable feature of this interface, it is not practical in the short term, since no existing authorisation infrastructure could easily comply with it, nor are we yet sure of the full set of requirements. Rather, this paper presents the basic minimum set of features that are needed to provide an initial plug and play functionality. Other features, such as a management interface, may be standardised in the future, whilst yet other features may continue to be met in an implementation specific manner.

The second half of this paper provides a brief introduction to the Security Assertions Markup Language (SAML) and says how each of the initial authorisation interface requirements can be met by either the basic SAMLv1.0 specification or by extensions to it. The paper concludes by anticipating the future standardisation effort that will be needed to completely specify an authorisation interface for the Grid.

1. Introduction

The Grid already has a strong authentication mechanism, in the form of a public key infrastructure (PKI). However, strong authentication on its own is not enough to ensure that registered users only have access to the Grid resources that they are entitled to. Users not only need to be authenticated, but also authorised to use specific Grid resources in a controlled manner. Thus the Grid needs a strong authorisation mechanism to complement its existing strong authentication mechanism.

Since users access resources at multiple locations on the Grid, and are themselves widely dispersed, the authorisation mechanism needs to be distributed. This will allow managers at different sites to authorise their users to use different Grid resources. However, controlling access to a particular Grid resource at a site needs to be within the remit of the local resource manager. He should be able to set the policy to dictate which users from which sites are allowed to access which local resources in which ways. Managers at different sites should be able to work autonomously, without needing to contact each other each time a new user needs to be authorised. Thus the managers need to trust each other to authorise their own users in an appropriate manner, and the authorisation infrastructure should support this. The lack of such a policy controlled authorisation infrastructure for the Grid is a significant impediment to the commercialisation of the Grid. It also makes it difficult for academic researchers to easily control access to Grid resources, both in controlling who can access the Grid resources under their local control, as well as authorising their own users to access remote Grid resources.

There are several policy based authorisation infrastructures now in existence, for example, CAS [CAS], VOMS [VOMS], PERMIS [Permis], and Akenti [Akenti]. Each of these has its own proprietary way of authorising users and setting the authorisation policy. Each of them supports trust establishment to a greater or lesser extent. Each of them has its own strengths and weaknesses. None of them has yet become the de-facto standard for the Grid authorisation infrastructure. Because they all have their own proprietary interfaces, and ways of working, it is difficult for Grid developers to easily switch between authorisation systems so as to be able to use the authorisation system that best meets the needs of their specific Grid application and its users.

It is for these reasons that the Globus development team first decided to specify a standard authorisation interface for the Grid. The first draft specification for an authorisation interface, based on the Security Assertion Markup Language [SAML], was published in February 2003 [Auth]. With significant UK input, this specification has now been substantially revised, and split into two parts, a requirements document [Req], and an enhanced interface specification [AuthRev]. The Globus team intends to implement the final specification in Globus Toolkit version 3. It is anticipated that authorisation infrastructure providers will also quickly implement this SAML Interface (PERMIS is already committed to doing so). This will allow Grid developers to plug and play different authorisation infrastructures so that they can find the one that best suits their needs.

The rest of this paper is structured as follows. Section 2 describes the requirements that have been identified for a distributed policy based Grid authorisation infrastructure. Section 3 provides a brief introduction to the SAML standard. Section 4 describes how the requirements of Section 2 have been mapped onto the current SAML specification. Section 5 concludes, and looks to possible future developments.

2 Authorisation Infrastructure Requirements

The ISO Standard 10181-3, Access Control Framework [ISO] specifies that the access control function can be split into two components, an application independent access control decision function (ADF), and an application dependent access control enforcement function (AEF). The ADF makes decisions about who can access which resources in which modes, and is governed by an access control policy. An AEF enforces these decisions on behalf of each application. The Internet RFC [FMWK] similarly defines a policy decision point (PDP) and a policy enforcement point (PEP). PDPs are functionally equivalent to ADFs, and PEPs are functionally equivalent to AEFs. Thus the first requirement for a Grid authorisation infrastructure, is to be able to separate the decision making from the decision enforcement, thereby allowing the authorisation decision making infrastructure to be Grid application independent.

Once we have agreed this separation of functionality, it is helpful to standardise the interface between these two components, so that (the AEF/PEP component of) Grid applications can utilise any ADF/PDP from any authorisation infrastructure supplier, and can choose the best one for their purposes. It is this interface that we plan to standardise

for Grid applications. When integrating an authorisation infrastructure into a Grid application, the decision making function (ADF/PDP) can either be embedded directly into the application, and accessed via an application programmable interface (API), or it can operate as a stand alone server, and be accessed via a message passing protocol, as in the COPS proposed standard [COPS]. If accessed via an API, then the application will need to be written in the same or compatible programming language as the ADF/PDP. Examples of authorisation APIs are the Open Group's Authorisation API [AZN], defined in the C language, and the PERMIS API, defined in Java. If the ADF/PDP operates as a stand-alone authorisation server and uses a standard message passing protocol interface, it is usually easier for applications to interface to it, and is programming language independent. It is this second scenario that the Grid authorisation interface primarily plans to address, although by using SAML (see later) the interface can be mapped to an API.

If the ADF/PDP runs as a stand alone authorisation server, then one needs to consider which types of client may access it. Should only application AEF/PEPs be allowed to access the authorisation server during the process of granting or denying a user's right of access, or should users (or their agents) be able to access the authorisation server directly, in order to determine the access rights that have been granted to them to particular targets. We think it is important that the interface should allow both types of client to access the authorisation server, but any given authorisation server implementation may limit the number of clients that it is willing to respond to. In the extreme this may be just one client, the AEF/PEP of the application being controlled by the authorisation server. There is no requirement for the interface specification to contain any guidance on how the authorisation server is to be configured with information about the clients that it is willing to respond to, as this will be a local configuration matter. However, two (of several) possibilities are:

- a. The authorisation server will only reply to requests where the client and the user are the same entity
- b. The authorisation server will only reply to requests where the client is a known and trusted AEF/PEP

Coupled to the above, it is clearly important that the authorisation server is able to authenticate the client, in order to determine who is making the authorisation request. Our current thinking is that the interface document should not specify how authentication is to be achieved but rather that this should be a local matter for the authorisation server implementer. Two possible authentication methods that could be supported are:

- a. The authorisation server might mandate that a mutually authenticated SSL/TLS session be established with the client
- b. The authorisation server might mandate that XML signatures be added to the authorisation decision requests.

An important issue to consider, is what should be the format of the decision returned by the authorisation server? Is a simple Boolean (where true=Granted and false=Denied) sufficient, or is a more flexible decision such as a score between 1 and 10 required? Our current thinking is that a simple Boolean is sufficient (but possibly qualified by

conditions – see later). If the authorisation server is responding directly to an AEF/PEP, then a simple Boolean is adequate. This allows the AEF/PEP to rapidly act upon the decision. However, if the client is a user, who then wishes to forward the response to an AEF/PEP, a simple Boolean is insufficient. The PEP will want to know exactly what has been granted or denied. Thus the decision response should optionally be able to contain full details of the request that is being granted or denied. Such a response is sometimes called a capability, since it says what the user is capable of doing.

Another issue to consider is how does the authorisation server gain access to the user's credentials that it will use to make its decisions. There are two modes of operation, termed (credential) pull and (credential) push in RFC 3281 [Aprof]¹. In credential pull mode, the authorisation server is merely presented with authenticated information about the user's identity, and the server must pull the user's credentials from some local or remote storage. It can then use these credentials in its subsequent decision-making. In credential push mode, the user's credentials are presented to the server by the client, so that the server does not have to go and retrieve them. If the server is operating in credential pull mode, then how does the server know where to get the credentials from? The server 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.

Given that both credential push and pull modes are supported, what should happen if there is a clash of expectations between the client and server? If the authorisation server is expecting to work in credential push mode, but the client expects the server to operate in credential pull mode and therefore provides no credentials, then the client should either be denied access or given the lowest access rights that are available to every authenticated user. If vice versa, the authorisation server may at its discretion either ignore the pushed credentials or use them in its decision-making. The interface should therefore allow the credential mode of use to be flagged by the client, and if credential pull mode is flagged, the client should be able to advise the authorisation server where to locate the credentials.

Similar to the above, how does the authorisation server gain access to the policy that controls it. The policy could be pre-configured into the server via some local configuration parameters, or via a management interface, or the client could pass the server the policy that is to be used at the time of decision-making. Clearly setting the policy has important consequences in trusting the decisions that are made, since a wrong policy will lead to wrong decisions being made. For this reason, our current plan is to separate policy management from the decision-making process, and only to concentrate on the latter initially. Future work will specify a management interface to the authorisation server that will allow a manager to dynamically update the policy that is to be used for decision-making. In the short term we expect it to be a local matter how the authorisation server is configured with the correct policy to be used.

¹ Note that RFC 3281 specifies them simply as push and pull modes of operation. We prefix this with "credential" in order to differentiate between the push and pull modes specified in RFC 2904, which have a different meaning.

One of the key features of the interface is to define how much information should be passed to the authorisation server in order for it to make its decisions. The following categories of information have been identified:

- 1) Information about the user making the access request. In the simplest case this could just be the authenticated name of the user. In more complex cases it could be an assertion about the name of the user, and details about the authentication service that authenticated the user, and the type of authentication that was performed.
- 2) Information about the operations/actions being requested by the user, including their operands e.g. write access to file Fred
- 3) Information about the target resource to be accessed e.g. C:\ or ftp://ftp.salford.ac.uk
- 4) Contextual and environmental information that is relevant to making the decision e.g. the time of day, or the number of resources currently being consumed by the user. This information is needed as some policies might stipulate conditions such as: access is granted only between 9am and 5pm, or access is granted up to 3 megabytes of storage.
- 5) The Credentials of the user. This will comprise such things as group memberships, roles and attributes of the user (which can all be regarded as attributes). The credentials will not only need to specify the types and values of the attributes, but also who assigned the attributes to the user, and how long the attributes are valid for. The latter information is especially important if the authorisation server is working in credential push mode, because the server will need to know that the user is not trying to use credentials that have already expired. If the credentials are long lived, and held by the user, then the authorisation server also needs to know where to retrieve revocation information from, so that it can check that the user is not trying to use revoked credentials.

All of the above information is usually needed for correct decision making. However, in some circumstances the environmental parameters may not be needed i.e. if the policy does not make use of them. As noted above, if the server is operating in credential pull mode, the user's credentials will not need to be passed with the decision request. Each of the above parameters (excluding the environmental ones) should be able to take a default value in the decision request. We attach the following meanings to the default values:

- 1) The user is anyone i.e. public access is being requested
- 2) The action can be anything i.e. all the rights that have been granted to the user
- 3) The target is all the targets that are protected by the authorisation server.
- 4) (there is no default for this)
- 5) The user has no specific credentials, only those default ones (if any) that have been granted to everyone should be used.

The client should be able to pass as little or as much information about the requested action as it wishes to the authorisation server. If too much information is passed (e.g. the current time is passed when the policy does not include temporal constraints, or a filename is passed when the user has access to an entire directory) then the server should

simply ignore the excess information when making its decision. If too little information is passed for the server to make an access decision, it may simply deny access.

Alternatively, at its discretion, and providing the client has not already been denied access from the information already provided, the server may return “Granted subject to” along with a set of conditions that must be fulfilled before access is granted E.g. Granted subject to the filename being Fred, or Granted subject to the time being between 9am and 5pm. It is then the responsibility of the AEF/PEP to evaluate these conditions before granting access to the user. If the AEF/PEP is unable or unwilling to evaluate these conditions, the client always has the option of issuing a new decision request and sending more information to the authorisation server. When conditional responses are returned, the AEF/PEP is actually being asked to perform some of the functionality of the authorisation server, i.e. evaluate some of the policy conditions that grant or deny access, and one could view this as a symptom of an incompletely implemented authorisation server. Therefore it should not be mandatory for an authorisation server to return the “Granted subject to” reply.

Finally, we need to consider how many stages are involved in the decision making process. Two different modes of operation have been identified - single step and multi-step decision-making. In single step decision-making, all the categories of information are passed in one message. This (potentially) contains enough information for the authorisation server to make its decision. In multi-step decision-making, the first step passes information about the user (categories 1 and 5 in the list above), and subsequent steps pass information about the user’s requested action(s) (categories 2, 3 and 4 above). Multi-step decision-making is an optimisation that allows the authorisation server during the first step, to collect together all the credentials of the user, and to validate them according to the policy. Expired or revoked credentials can be discarded, as can credentials that are not recognised by the policy e.g. having a credential asserting membership of a university faculty when the policy grants access according to professional qualifications. The collected valid set of credentials are then used in subsequent decision making steps. The decision making step can be repeated as often as necessary, as the user attempts to invoke different operations/methods on different resources. Multi-step decision making can be implemented by either a state based or state-less authorisation server, and the interface should support both types of server. A state based server will cache the collected valid set of credentials internally and use them when the client requests subsequent authorisation decisions to be made. A state-less authorisation server will return the collected valid set of credentials to the client, as assertions made by itself, and the client can cache these and use them in subsequent decision making steps (rather like web browsers store cookies for web servers).

3. An Introduction to SAML

The Security Assertion Markup Language (SAML) is a specification issued by OASIS, an industry led standards creating consortium. SAML is a general purpose language that allows different types of security assertions about subjects to be passed between clients and servers, encoded as XML messages. The language is infinitely extensible and allows any type of assertion to be defined, although three standard types of assertions about subjects are currently specified in SAML v1.0: authentication assertions, attribute

assertions and authorisation decision assertions. An authentication assertion states how the subject was authenticated, an attribute assertion states which attributes have been assigned to a subject, and an authorisation decision assertion states which access to which targets has been granted or denied to the subject. Each assertion is accompanied by the name of the issuer of the assertion, the date and time the assertion was issued, an assertion ID (for ease of subsequent reference) and the version of SAML that the assertion conforms to. Each assertion may optionally be accompanied by some conditions (which must be obeyed by the assertion user), by some advice (that can be ignored by the assertion user), and by the digital signature of the issuer.

SAML XML assertion messages can be transported by a variety of means. They can be transported “as is”, for example by passing them as fields in an API, embedding them in email messages or carrying them straight over TCP/IP. However, the SAML specification defines a request/response protocol for requesting SAML assertions and embedding them in SAML responses. A SAML request can be either an authentication query, an attribute query or an authorisation decision query, and it comprises: the type of query, a unique request ID, the date and time the request was issued, the type of assertion responses the requestor would like to be returned to the query (called Respond With parameters), the digital signature of the requestor (optional) and the version of SAML this message conforms to. An authentication query asks “What assertions containing authentication statements are available for this subject?” An attribute query asks “Can you return the requested attributes for this subject?” An authorisation decision query asks “Should these actions on this resource be allowed for this subject, given this evidence?” A SAML response comprises: zero or more SAML assertions, a status code (indicating if the request was successful or not), the unique ID of the original request, a unique ID for this response (optional), the time the response was issued (optional), the intended recipient of the response (optional), the digital signature of the responder (optional) and the version of SAML this message conforms to. The SAML request/response protocol can then be mapped onto existing protocols such as TCP/IP or HTTP via SAML Bindings and Profiles.

The SAML Bindings specification [Bind] describes one such binding. It defines how the SAML request/response protocol can be carried as SOAP messages over HTTP. It also gives guidance for how the SAML protocol can be carried over other transport mechanisms, leaving implementers free to define the mappings for the protocol of their choice.

4. The SAML Grid Authorisation Interface

The SAML protocol Authorisation Decision Request forms the basic message for issuing a request to the authorisation server. It already has most of the fields that are needed to pass the information categories identified in section 2, and it is then a question of mapping them as described in section 4.1. below. However, some of the requirements cannot easily be mapped into existing SAML elements, and so new data elements need to be defined. These are described in section 4.2. The SAML protocol Authorisation Decision Response forms the basic message for returning granted/denied/granted-subject-to responses to the client, and this is described in section 4.3. However, this message is

not able to send simple decision responses, but is only capable of returning the entire Authorisation Decision Request message in the response. Consequently a new simple Decision message needs to be defined and this is described in section 4.4. Finally, SAML in its basic form is not capable of supporting multi-step decision making, so the extensions that have been defined to support this are described in section 4.5.

4.1 The SAML Authorisation Decision Request

As stated earlier, this message comprises: an authorisation decision query, a unique request ID, the date and time the request was issued, the type of assertion responses the requestor would like to be returned (called Respond With parameters), the digital signature of the requestor (optional) and the version of SAML this message conforms to. The authorisation decision query comprises: the name of the subject, the URI of the resource, the actions for which authorisation is being requested, and evidence to support the request.

Looking at each of the authorisation decision query parameters in detail, SAML supports subject names in the form of X.509 distinguished names. This is the format recommended for use in the Grid authorisation interface, since every Grid user has a distinguished name in his public key certificate. If the request is for (default) public access, rather than a specific user, then the distinguished name should be set to null (implying the root of the directory information tree).

If the resource is a Grid service, then the resource URI should contain the Grid Service Handle (GSH) of the service as described in [OGSI]. If the client wants to learn about the access rights of the user to all resources known to the authorisation server, then the following default URI has been defined: <http://www.gridforum.org/ogsa-authz/saml/2003/06/resource/any>

SAML actions are composed of a string describing the operation or method to be authorised, and a URI specifying the namespace of the action. The SAML specification already defines several sets of actions and their associated namespaces, including: Get/Head/Put/Post for the HTTP protocol, the standard Unix file permissions (read, write, execute etc.), and a set of generic operations. The latter are:

Read/Write/Execute/Delete/Control and the associated namespace URI is `urn:oasis:names:tc:SAML:1.0:action:rwdc`. If this standard set of actions is insufficient, then Grid application developers will need to specify their own set of operations for their Grid applications, along with associated namespaces. If authorization for all possible actions is being requested, then the action string should be set to an asterisk (*), and the following namespace has been defined in [AuthRev]:

<http://www.gridforum.org/namespaces/2003/06/ogsa-authz/saml/action/wildcard>

The SAML evidence field is used to hold the user's credentials. However, we had a problem when trying to fit both the credential push and pull modes into the SAML protocol. When operating in credential push mode this element can obviously contain the user's credentials. If the initiator does not have any credentials (for example, if default or

public access rights are being requested) then the evidence field can be empty². But how should the client signal to the authorisation server that it wishes to operate in credential pull mode? An empty evidence field can be used to signal pull mode at the discretion of the server, but how can the client direct the server to a credential repository? An extension to SAML was needed for this, and so we defined the Reference statement (see section 4.2).

The final issue to be solved, was how can the client pass environmental parameters to the authorisation server, since SAML does not have a specific field for this? We solved this by considering the environmental parameters to be additional evidence, set by either the Grid application (e.g. the amount of storage already consumed by the user) or some other trusted entity (e.g. a trusted time source for time of day). Consequently, the environmental parameters can either be pushed to the authorisation engine, or they can be pulled from somewhere. Thus the SAML evidence field can contain additional elements holding environmental parameters, or Reference statements to where the environmental parameters can be found. Alternatively, the authorisation server can be configured with a local list of where to get this information from.

Turning now to the Authorisation Decision Request message in which the authorisation decision query is embedded, most of the parameters are fairly straightforward to complete as per standard SAML guidelines, and do not need any special rules for Grid use. The only parameter that has been specifically tailored for Grid use is the RespondWith parameter. This informs the authorisation server what sort of SAML response the client would like. In standard SAML, this will always be an Authorisation Decision Assertion Response, but this message has its limitations - see section 4.3. If the client would like a full authorization decision assertion to be returned then RespondWith is set to the value *Authorization*. If however, the client would like a simple decision response to be returned (see section 4.4) then RespondWith is set to the value *Decision*.

4.2 Extensions to the SAML Request Message

We noted in 4.1 above, that in the credential pull mode of operation, the client may wish to advise the authorisation server where to find evidence (either user credentials or environmental parameters). We have defined the SAML Reference statement to fulfil this function. The <ReferenceStatement> element supplies a statement that the designated attributes associated with the specified subject can be obtained from the referenced URI. Thus if the user was the holder of X.509 attribute certificates held in an LDAP server, the Reference Statement URI would point to the LDAP server, the subject name would be the LDAP DN of the subject, and the attribute name could be set to an asterisk '*' (meaning fetch all attribute certificates). Alternatively, if the client wished to inform the authorisation server of the locality of a trusted time stamping authority (TSA), the Reference Statement URI would point to the TSA, and the attributes and subject elements would be set to their default values ('*' and null respectively). When embedded in an Evidence element, the issuer element is set to the name of the client.

² To be more precise, there will be no evidence assertions in which the subject name is that of the user.

4.3 The SAML Authorisation Decision Response

As stated earlier, the SAML Authorisation Decision response comprises: zero or more authorisation decision assertions, a status code, the unique ID of the original request, a unique ID for this response (optional), the time the response was issued (optional), the intended recipient of the response (optional), the digital signature of the responder (optional) and the version of SAML this message conforms to. Each authorisation decision assertion states which actions to which targets has been granted or denied to the subject, given the attached evidence and optional conditions. In short, the authorisation decision request is repeated in the authorisation decision response, even though the messages can be linked via their unique IDs. Each assertion is accompanied by the name of the issuer of the assertion, the date and time the assertion was issued, an assertion ID, the version of SAML that the assertion conforms to and the optional digital signature of the issuer.

The first thing to note is that the standard Authorisation Decision Response message is potentially huge, and there is no way of returning a simple Boolean decision to the request. Consequently a simplified Authorisation Decision element was defined (see below). The client can indicate if a full authorisation decision assertion or a simple authorisation decision is to be embedded in the response, by appropriately setting the RespondWith parameter on the request.

The second thing to note is that the SAML simple decision type has three possible values: Permit/Deny/Indeterminate, but this does not include Granted-subject-to. We have chosen to use the Indeterminate SAML value to mean Granted-subject-to, and if this value is present, then the response must contain one or more condition statements.

4.4 Extensions to SAML Responses

The Authorisation Decision response has been defined as a simple way of passing a Boolean response to the client. It only comprises the simple decision type (with the additional restriction that Indeterminate is not allowed), and it is linked to the corresponding Authorisation Decision Query via the unique request ID. Note that in most cases simple authorisation decision responses will only be requested by Grid application AEF/PEPs, since a Grid user is unable to relay this response to a Grid application as it does not say what the user is allowed to do (unlike the full authorisation decision assertion).

4.5 SAML extensions to support multi-step decision making

The basic method used to drive multi-step decision making is use of the RespondWith parameter of the SAML Authorisation Decision Request message. If the client sets RespondWith to the value *Attribute* it signals the first step of multi-step authorization to the server. The authorisation server will then ignore the resource and action fields of the request (and the evidence field will be empty). The authorisation server may be state based or state-less. The only requirement is that it returns an attribute assertion to the client, the contents of which are currently undefined and server implementation specific. The client must then return the same attribute contents in the evidence field of subsequent authorisation decision request messages. In the subsequent requests, the client can ask for

either simple authorisation decisions or authorisation decision assertions to be returned by setting RespondWith to either *Decision* or *Authorization* as before.

5. Conclusions and Future Work

It is clear from section 4, that SAML v1.0 only standardises the basic protocol necessary for supporting a fully functional and flexible authorisation interface. Several extensions had to be defined in order to get the richness that we needed. We are not the only researchers to discover this. The OASIS XACML [XACML] working group have similarly noted many of the same deficiencies as Grid researchers. Consequently Grid and XACML researchers are now working together to unify the format and semantics of the extensions that they have identified, and are relaying these to the OASIS SAML working group. The SAML group are in the process of defining SAMLv2.0, and plan to publish this in late 2003 or early 2004. We hope that all of the extensions that we have identified will be incorporated in the base SAML v2.0 specification, and then no Grid specific SAML extensions will need to be defined.

One slight setback has been the recent publication of SAML v1.1. This specification deprecates the use of the RespondWith parameter defined in SAML v1.0, which is a key feature of the Grid use of SAML for its authorisation interface. We are currently working with the XACML and SAML researchers to determine the best way forward here.

Once we have a fully functional SAML protocol, this is actually only the start of the standardisation work for an authorisation interface. It will allow Grid application developers to plug and play different authorisation infrastructures that support SAML, but they must plug and play entire infrastructures, and not simply different components of different infrastructures. In order to do the latter, each of the fields that can be passed in the SAML messages will need to be standardised. Then each of the authorisation infrastructures will use standard data structures and syntaxes for passing user credentials, authorisation policies, condition statements and environmental variables. There are some existing candidates for each of these. XACML defines a standard authorisation policy, using XML, and this policy standardises the way of passing condition statements and environmental variables. ITU-T X.509 (2001) [X509] defines a standard for user credentials, called attribute certificates, and these are similar to existing X.509 public key certificates. Proposed Internet Standard RFC 3275 [Xsig] defines a standard way of adding digital signatures to XML documents.

It will be interesting to see how things develop in the coming months. Watch this space.

References

[Akenti] Johnston, W., Mudumbai, S., Thompson, M. "Authorization and Attribute Certificates for Widely Distributed Access Control," IEEE 7th Int Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Stanford, CA. June, 1998. Page(s): 340 -345 (see also <http://www-itg.lbl.gov/security/Akenti/>)

[Aprof] Farrell, S., Housley, R. "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002

[Auth] Von Welch, Frank Siebenlist, Sam Meder, Laura Pearlman. "Use of SAML for OGSA Authorization" 15 Feb 2003.

[AuthRev] Von Welch, Frank Siebenlist, David Chadwick, Sam Meder, Laura Pearlman. "Use of SAML for OGSA Authorization", June 2003, Available from <http://www.globus.org/ogsa/security/>

[AZN] The Open Group. "Authorization (AZN) API", January 2000, ISBN 1-85912-266-3

[Bind] P. Mishra et al., "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)". OASIS, April 2002. See: <http://www.oasis-open.org/committees/security/docs/cs-sstc-bindings-00.pdf>

[CAS] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. "A Community Authorization Service for Group Collaboration". Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[COPS] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry. "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[FMWK] Yavatkar, R., Pendarakis, D. and R. Guerin, "A Framework for Policy-Based Admission Control", RFC 2753, January 2000.

[ISO] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework"

[OGSI] Foster, I., C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[Permis] D.W.Chadwick, A. Otenko "The PERMIS X.509 Role Based Privilege Management Infrastructure". Future Generation Computer Systems, 936 (2002) 1–13, December 2002. Elsevier Science BV.

[Req] Von Welch, Frank Siebenlist, David Chadwick, Sam Meder, Laura Pearlman. "OGSA Authorization Requirements", June 2003. Available from <http://www.globus.org/ogsa/security/>

[SAML] Phillip Hallam-Baker et al. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)". OASIS, 19 April 2002. See <http://www.oasis-open.org/committees/security/>

[VOMS] "VOMS Architecture v1.1," http://grid-auth.infn.it/docs/VOMS-v1_1.pdf, February 2003.

[XACML] "OASIS eXtensible Access Control Markup Language (XACML)" v1.0, 12 Dec 2002, available from <http://www.oasis-open.org/committees/xacml/>

[Xsig] Eastlake, D., Reagle, J., Solo, D. "(Extensible Markup Language) XML-Signature Syntax and Processing" RFC 3275, March 2002

[X509] ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Public-key and attribute certificate frameworks