Job-Centric Security Model for Open Collaborative Environment

Yuri Demchenko
Universiteit van Amsterdam
demch@science.uva.nl

Cees de Laat
Universiteit van Amsterdam
delaat@science.uva.nl

Andrew Tokmakoff
Telematica Instituut
Andrew.Tokmakoff@telin.nl

Leon Gommans
Universiteit van Amsterdam
lgommans@science.uva.nl

Bas Oudenaarde
Universiteit van Amsterdam
oudenaarde@science.uva.nl

Martin Snijders
Telematica Instituut
Martin.Snijders@telin.nl

ABSTRACT

This paper describes the design and development of a flexible, customer driven, security infrastructure for Open Collaborative Environments. The experiences were gained within the framework of the Collaboratory.nl project. The work is based on extended use of emerging Web Services and Grid security technologies, combined with concepts from the generic Authentication Authorization and Accounting (AAA) authorisation framework. Basic CNL use cases and functional security requirements are analysed to provide motivation for the proposed Job-centric security model. This model describes access control and user- and resource management. The proposed Job-centric approach uses a Job description as a semantic document that is created on the basis of the signed order (or business agreement). It contains all the information required to run the experiment and also to create/manage the virtual Job-based associations of users and resources. The proposed trust relations analysis explains the use of trust anchors in the Job-centric security model. In addition, the paper provides implementation details of using XACML and SAML for Authorisation assertions and messaging, based on the current CNL implementation.

KEYWORDS: Open Collaborative Environment, Job-centric security model, authorisation framework, RBAC, SAML, XACML

1. INTRODUCTION

Many modern research areas (e.g. the process industry) rely on advanced laboratory equipment, such as electron microscopes, mass spectrometers, equipment for surface analysis , and other analytical equipment. Effective use of this equipment during experiments and for production work requires complex infrastructure and the involvement of many specialists that may be distributed and span multiple organisations. Emerging Computer Grids and Web Services technologies provide a sound basis for extending Groupware, which has traditionally been used for collaborative applications to build a virtual collaborative environment. Such virtual laboratories offer the same possibilities as a traditional laboratory, however they also enable laboratory staff to utilise the equipment and expertise of third parties. Security services provide a reliable and secure operational environment that is capable of managing customers' and providers' resources. Protection of privacy and confidentiality is of particular importance when different parties share the same equipment.

This paper presents the experience of designing and developing an open, flexible, customer-driven security infrastructure for open collaborative applications in the framework of the Collaboratory.nl[1] project (CNL). The work is largely based upon extended use of emerging Web Services and Computer Grid security technologies and the generic AAA authorisation framework [1, 2, 3, 4].

Collaborative applications require a sophisticated, multi-dimensional security infrastructure that manages secure operation of user applications between multiple administrative- and trust domains. Typical Open Collaborative Environment (OCE) use cases requires that the collaborative environment:

- is dynamic since the environment can potentially change from one experiment to another,
- may span multiple trust domains,
- can handle different user identities and attributes/privileges that must comply with different policies (both experiment and task specific).

[1] http://www.collaboratory.nl/

Managing access based upon role-assigned privileges and policy enforcement are addressed in many collaborative and Computer Grids projects. The majority of known solutions and implementations [5, 6] use widely recognised Role-based Access Control (RBAC) [7] models as a general conceptual approach, and XACML [8] as an implementation basis. The current Grid Security Infrastructure and Authorisation framework evolved from using proprietary solutions such as the Community Authorisation Service (CAS), toward the use of a XACML-based Policy Management and Authorization Service, as seen in for the most recent Globus Toolkit 4.0 release [5, 6, 9, 10]. Although providing a good example of addressing similar tasks, current Grid based solutions don't provide all of the required functionality for the OCE. Their deep embedding into parallel task scheduling mechanisms prevents distributed execution of dissimilar computational tasks/jobs. The OCE is less coupled and mostly concerned with the allocation and execution of complex experiments on the equipment that for most use cases require human control and interaction during experiment.

Collaborative tools like Chef[2], initially designed for online educational course management, can provide most of the necessary functionality for the creation of a collaborative environment. However, this environment needs to be extended such that it can be integrated with other stages and components of the collaborative organisation managing the experiment stages. These stages include the initial stage of order creation and the main experimental stage that requires secure access to the instrument or resource.

To address these specifics, the OCE security architecture proposes a novel Job-centric approach, which is uses the Job description as a semantic document, created on the basis of a signed order (or business agreement) [11, 12]. The document contains all the information required to run the analysis, including the Job ID, assigned users and roles, and a trust/security anchor(s) in the form of the resource and additionally the customer's digital signature. In general, such approach allows binding security services and policies to a particular job and/or resource.

This paper is organized as follows: Section 2 presents basic OCE use cases, the required security functionality and also introduces the proposed Job-centric security model. Section 3 describes the operation of the OCE security system which is built around the Job description as a semantic document defining security context for OCE security services operations. Section 4 provides more details about policy-based access control in the OCE, and discusses issues associated with combining multiple policies and multi-level access control enforcement. In addition, implementation suggestions are provided based upon CNL practical experience. Section 5 attempts to formalise the trust relations evident in an open distributed access control system, using the Job-centric security model and RBAC.

Finally, section 6 provides additional implementation details, describing the CNL Authorisation service which combines RBAC functionality with the generic AAA Authorisation framework. The CNL Authorisation service provides a good example of using XACML and SAML standards for Authorisation assertions and messaging.

The proposed approach and solutions are being developed to respond to both common and specific requirements of the CNL and EGEE[3] projects. The approach and can also represent a typical OCE use case for the general Web Services and OGSA Security framework. It is expected that other project may stand to benefit from this work, as it proposes a general approach and common solutions for the security problems found in OCEs.

2. GENERAL OCE SECURITY REQUIREMENTS AND PROPOSED JOB-CENTRIC SECURITY MODEL

Security services are defined as the component of the OCE middleware that provides a secure infrastructure and environment for executing OCE tasks/jobs. Generally speaking, security services can be added to an already existing operational architecture, however current industry demand for very secure operational environments requires that the Security architecture is developed as an integral part of the system design. There should be also the possibility to define a security services profile at the moment of a system service invocation defined by a security policy.

For the purpose of analysing the required security functionality, the OCE use cases can be divided into two groups; simple security interactions and extended interactions. In a simple interaction use case, the major task is to securely provide remote access to instrument(s) that belong to a single provider. For this case, the remote site or the resource owner can provide few onsite services and allow distributed user groups. An extended use case must additionally allow distributed multi-site services, multiple user identities and attribute providers, and distributed job execution. In its own turn, multiple trust

[2] http://www.chefproject.org/

[3] http://public.eu-egee.org/

domains will require dynamic creation of user and resource federations/associations, handling different policies, specific measures for protecting data confidentiality and user/subject privacy in a potentially uncontrolled environment.

In both cases there is a need for the following functionality:

- fine-grained access control based upon user/subject attributes/roles and policies defined by a resource.
- privilege/attribute management by a designated person holding responsibility for a particular experiment or job.
- customer-managed/controlled security environment with the root of trust defined by a user/subject (or their private key). The security environment should also allow secure isolation of the execution of customer tasks on OCE facilities protected by the customer-controlled security key/credentials.

The above-listed requirements may be successfully addressed within the proposed job-centric approach to security services provisioning. Procedures in the OCE include two major stages as part of accepting and executing the order: negotiation and signing of the order (business part), plus performing the experiment (technical part). The Job description, as a semantic document, is created based upon the signed order and contains all information required to perform the experiment on the OCE infrastructure. The job description includes a Job ID, Job owner, assigned users and roles, and trust/security anchor(s) in the form of both resource and customer digital signatures. This kind of Job Description can be used as a foundation for creating a Virtual Organisation (VO) instance, as an association of designated users and resources which supports all "standard" security constructs such as users, groups, roles, trust domains, designated services and authorities [2, 13]. Figure 1 illustrates the structure of a Job Description and also its relation to other OCE components and security services.

The Job Description must include (or reference) the Job policy, which defines all aspects of the user, resource and trust management that should be take into account when executing the job. This policy should define the following items:

- trusted users, VO's, resources and in general, trusted credentials (or trusted Certification Authorities);
- delegation policy and identity federation/mapping policy (additionally);
- privileges/permissions assigned to roles;
- credit limits and conditions of use;
- confidentiality and privacy requirements;
- Job access control or authorisation policy.

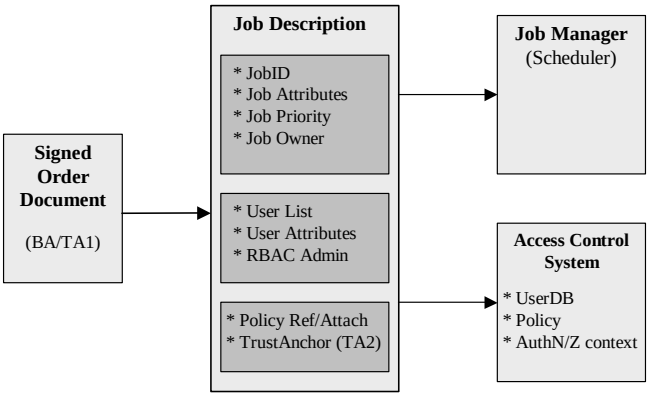

Figure 1. OCE Security built around a Job description

It is important to note that a Job policy may be combined with the Resource admission policy and in practice should not be more restrictive than the Resource policy. Otherwise, the Job security management service may reject some resources based upon Resource policy evaluation as a procedure of mutual authorisation.

Such a job-centric approach gives organizations complete flexibility in the creation of their security associations and services for their specific tasks or applications.

3. OCE SECURITY SYSTEM OPERATION

Each OCE has a need for basic security services: authentication (AuthN) and single-sign-on (SSO), policy based authorisation (AuthZ), information and data confidentiality and integrity, non-repudiation and privacy. Security services may be bound to and requested from any basic OCE service using a standard request/response format. Use of security services must be specified by the policy that provides a mapping between a request context (e.g., action requested by a particular subject on a particular resource) and resource functionality and access permissions. A binding between (basic) services and security services can be defined dynamically at the moment of service deployment or invocation using existing Web services and XML Security technologies for associating/attaching security services and policies to the service description [14, 15, 16].

Figure 2 illustrates the relations and interactions between major entities and processes in our Job-centric security model, including the actors/principals of the customer site and the services/(semantic) documents/entities of the resource site. For the purpose of trust analysis in section 5, the modules are grouped by shared trust relations in relation to the resource that is considered to be the root of trust for the model.

The initial information required for proper operation of the AuthN/AuthZ system should be provided in a job description (JobDescr) that binds job attributes, user information and established security/trust relations between the customer and the provider. This approach to building security services in the OCE, defined as a job-centric, provides the perceived benefit of decoupling security services from the application specific components, thereby simplifying the construction of the scalable distributed security infrastructure.

The JobDescr artefact is created as the result of customer and provider negotiation and an agreement that can provide the so-called business and/or trust anchor (BA/TA). During operation, security services will:

1) retrieve user information and roles from the JobDescr and put them into the UserDB;
2) retrieve job attributes to reference or define the policy of the resource access;
3) use TA or BA to verify or sign all future security (or financial) related attributes, claims, tokens and credentials.

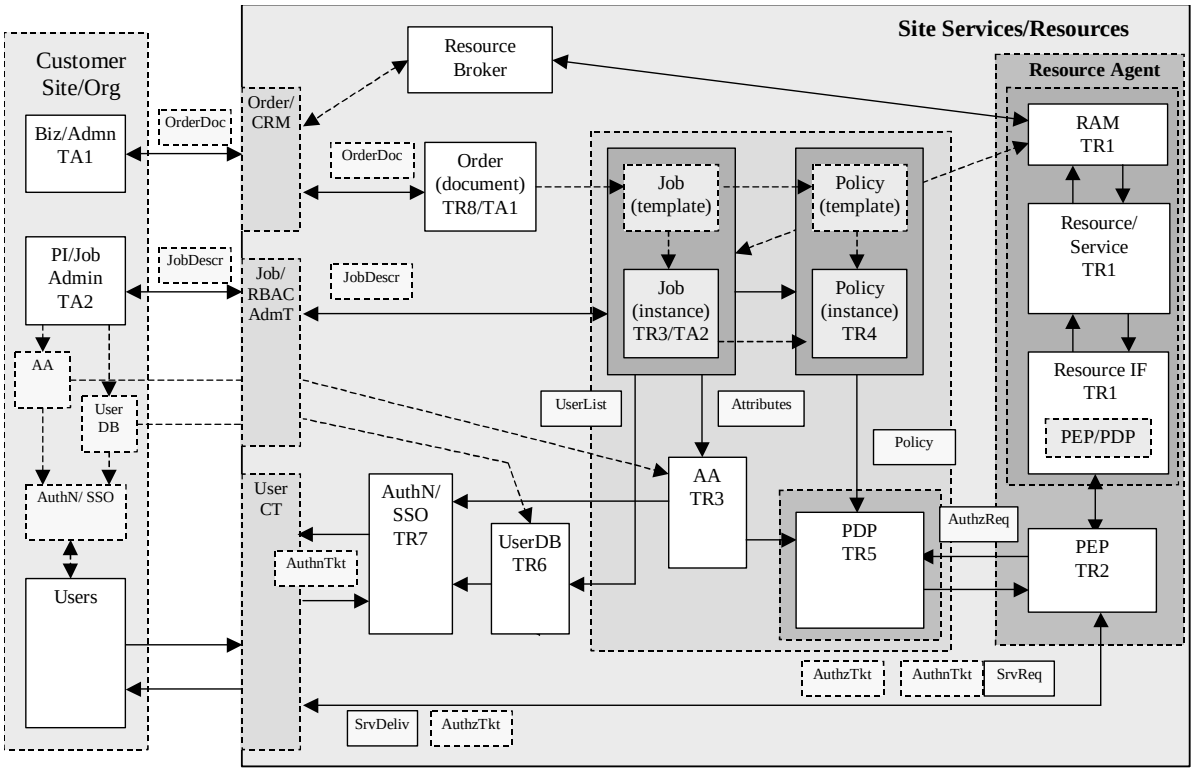

Figure 2. Major interacting components and entities in the Job-centric security model

Interaction with the user is provided via the User Collaborative Tools (UserCT) and their interaction with the instrument, via the Resource Agent. User authentication is requested by UserCT, user authorisation is enforced by the Resource Agent. The UserCT and Authentication services (AuthN) may provide a SSO (Single-Sign-On) functionality to provide a single user logon for a particular domain, defined by a business or trust agreement.

The Principal Investigator (PI) or Job/order owner may possess the RBAC administrative functions (privileges) that allow him/her to create and/or modify user accounts and assign roles/privileges for a particular job/experiment via the Job/RBAC Administration tools.

To allow user access to the resource, the Resource Agent requests (via the Policy Enforcement Point (PEP)) an authorisation decision from the Policy Decision Point (PDP). It is the PDP that evaluates the authorisation request against the policy defined for a particular job, resource and user attributes/roles. The access policy is defined by the resource owner and stored in the policy repository. During policy evaluation, the PDP may request specific user attributes from the Attribute Authority (AA) and, additionally, user identity confirmation from the AuthN service.

The resource interacts with the OCE via the Resource Allocation and Manager (RAM) and via the Resource interface (Resource IF) which may contain an internal PEP/PDP that controls the resource access, based upon the internal resource's usage policy and conditions.

4. POLICY BASED ACCESS CONTROL USING GENERIC AAA FRAMEWORK

A typical access control use-case may require combination of multiple policies and multi-level access control enforcement which may take place when combining newly-developed and legacy access control systems into one integrated access control solution. Figure 3 illustrates a typical RBAC authorisation model that implements the combined pull-push model of the generic AAA Authorisation framework in the interest of performance optimisation.

The diagram also explains how combining of multiple policies can be achieved, via PEP chaining/sequencing and/or PDP nesting/recursion. The proposed approach retains the integrity of the single/combined policy based decision. Thus, when the PDP evaluates a request from the PEP, it can call for external evaluation of some policy components but makes its own final decision, returning it to the calling PEP which acts as a gateway for the initial request.

The Requestor requests a service by sending a service request ServReq to the Resource's PEP providing as much or as little information about the Subject/Requestor, Resource, Action, and additionally Environment as it decides necessary according to the applicable authorisation model and (should be known) local policies.

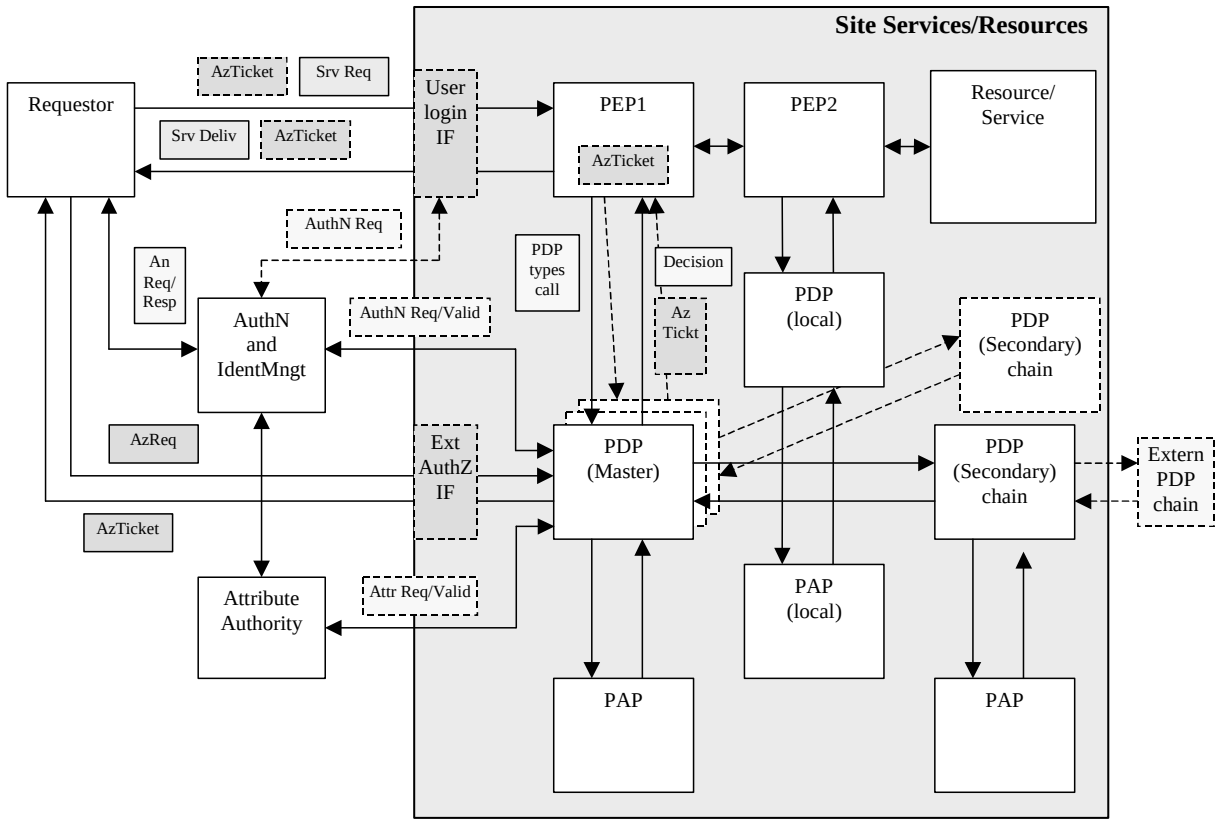

Figure 3. Major components of the site Authorisation service (RBAC and combined pull-push model)

In a simple scenario, the PEP sends the decision request to the (designated) PDP and after receiving a PDP decision, relays a service request to the Resource. The PDP identifies the applicable policy instance and retrieves it from the Policy Authority Point (PAP), collects the required context information and evaluates the request against the policy. During this process, it may need to validate the presented credentials locally, based upon pre-established/shared trust relations, or call external Authentication and Attribute Authorities.

In an open heterogeneous environment, the PEP may receive requests that use different formats and semantics (namespaces) and refer to different policies and/or policy repositories. In this case, the PEP should have the possibility to relay a decision request to the proper PDP type which is able to handle the decision request. It is essential that a request is evaluated in total and a decision is made by a single PDP, which however can make subsequent calls to external PDP's to evaluate some request components and process their decisions as components of the general policy evaluation process. The PDP that makes a final combined decision can be referred to as the master PDP and it needs to have mechanisms in place to preserve the integrity of its final combined decision.

Existing (open) policy expression formats such as XACML [8] and AAA [17], provide mechanisms for a particular policy instance to refer to another policy instance. A complex/combined policy can be created by a PAP on PDP policy request, or processed by the PDP by requesting the required policy components during the request evaluation.

As a trade-off of being open through the use of separate access control components and open standards, the solution presented above has known performance concerns, namely that requesting a remote PDP decision involves the use of time and resource hungry components such as building a remote SSL/TLS connection, XML message parsing, possible remote policy request and PDP/AuthZ service invocation. In total, this may cause a delay ranging from 40 to 800 milliseconds. This trade-off can be resolved by combining the pull and push operation models. Since the decision is made by the PDP, the authorisation ticket AuthzTicket can be issued and used in the subsequent similar or repeat action requests during the ticket's validity period. The AuthzTicket can be obtained via the PEP during the first access request or request directly from the PDP via an external AuthZ interface prior to sending service request.

The scenario described above is a basic one, but it requires that both the Requestor and the Resource services know (either explicitly or implicitly) and share the following security context: access control policy, namespace(s) and semantics, established trust relations, - which should be established prior to the security services initiation. Consequently, the following implementation suggestions should be considered:

1. Every PEP in the chain of policy enforcement should take care of the whole request evaluation/enforcement by calling to a single (master) PDP. The PEP should not do the combination of multiple decisions. Only one PDP should provide a final decision on the whole request. However, the PEP may have the possibility to request different PDP types, based upon the request semantics/namespace and referred policy.

2. It is suggested that in general (and to have the possibility of combining the pull and push AuthZ models for the performance reasons), the PEP should understand and have a possibility to validate the AuthzTicket issued by the trusted PDP or AuthZ service. For this purpose the Requestor may request, the PDP may issue and the PEP may relay the AuthzTicket back to the Requestor. The AuthzTicket issued by the PDP should have an associated validity period, usage restriction and should also contain information about the decision and the resource. For further validation of the AuthzTicket, the PEP may cache the ticket locally to further speed-up the validation procedure.

WS-Policy and WS-PolicyAttachment provide mechanisms to dynamically link security services to the OCE basic services [15, 16]. This can be done by associating/attaching the policy definition or reference to the service description in the WSDL format in the following way:

- the central point of the policy attachment is the service description in a form of a WSDL file, which contains a definition of the portTypes, available services and messages format. Attaching a policy to WSDL means that the policy reference can be added to any of the WSDL elements.
- interacting services will fetch policy document and apply restrictions/rules to elements, which declared policy compliance requirements; this may apply to both service request and response or service delivery.

In this case, security services may be added dynamically to a requested (basic) service instance at the time of its invocation.

5. TRUST RELATIONS IN DISTRIBUTED AAA INFRASTRUCTURE

This section provides a high-level analysis of trust relations for the general Job-centric security model discussed in section 3 (see also Figure 2). The analysis is intended to provide recommendations for the required trust and policy authority relations, including relations between Resource, Policy and PDP trust domains and authorities, and requirements on key/credential distribution for the OCE security architecture.

For the purpose of analysis, Figure 2 combines the main components of the OCE access control infrastructure into groups that have the same level of trust and/or authority in respect to the decision-making process and its context. It is assumed that in the resource access control model, the root of trust belongs to the resource.

The OCE/CNL Job-centric model uses one or two trust anchors (TA's) shared between the Customer and the Resource sites:

- TA1 – is contained in the signed Order created at the business negotiation stage and is optional.
- TA2 – included in the JobDescription that is created by the PI or Job owner and contains all the necessary context information for configuring the OCE security services instances. TA2 is a mandatory element of the discussed model.

For convenience, all components of the resource site are assigned credentials, their trust paths to the root of trust (defined by the Resource) are marked as **TRn**, where **n** is an integer. Using credentials path semantics proposed in [18], the following trust/credentials chain and delegation are considered between major modules/objects, where a credential is identified by the issuer or semantic document as a prefix and an attribute/role as a suffix.

```
User => HomeOrg.staff
      => Job.members (TA2)
            => Member.roles
                  => Role.permissions
```

The expression above can be read as follows: The user will have a final permission (to do an action), if s/he has a credential from the HomeOrg with attribute "staff", s/he is contained in the "members" list of the Job description, s/he is assigned a role in the members attribute list (may be a part of the Job Description or AA repository), and finally the user's designated role is assigned a "permission" to do an action. The final mapping between the roles and permissions is provided by the policy.

It is suggested that if a chain of delegation/credentials spans different trust domains, the trust anchor should be placed in the joint point. TA2 in our example is bound to the semantic document Job Description that can be easily shared between the Resource and the customer.

Using the above semantics, the process of obtaining the required permissions to perform the requested action by the user can be described in the following form:

```
User
=> AuthN(HomeOrg.staff, Job.members)
=> AuthZ(Member.roles, Policy.permissions)
=> Resource.permissions
```

The above analysis is a very initial attempt to tackle the problem of formalising trust relations in distributed access control systems. More research will be needed to propose a more structured approach and solution. With further analysis it should be possible to provide recommendations for key management and policy management in the proposed Job-centric security infrastructure.

To become a shared trust anchor between the resource and the customer domains, the Order (TA1) or Job Description (TA2) must contain mutually signed credentials/certificates. Although the main PEP operation will assume a pull authorisation decision request to the trusted PDP, in general it may accept the AuthzTicket from an external PDP belonging to the trusted domain.

6. USING SAML AND XACML FOR AUTHORISATION ASSERTIONS AND MESSAGING

The proposed Job-centric security model is being implemented in the CNL Authorisation service. CNL uses a proprietary Job Description format, which in the future can be mapped to two related formats: WS-Agreement [19] and Job Submission Description Language (JSDL) [20] being developed within the OGSA Framework. The CNL Authorisation service uses the standard XACML messaging format for PEP-PDP communications and the XACML policy format for policy exchange and combination. SAML [21, 22] is used as a security assertions format and in particular for the CNL Authorisation ticket (CNLAuthzTicket).

The Request message consists of three mandatory elements Subject, Resource, Action (the so-called Target triad), and optionally may contain the Environment element. The Subject element consists of the SubjectID, SubjectConfData, Role and JobID sub-elements. The Resource element contains a ResourceID sub-element that

specifies the CNL resource or instrument, and may contain multiple ResourceAttribute sub-elements that may define a resource subsystem or content-related attribute. The Action element contains only one sub-element which is ActionID. It will be also possible to request multiple actions, however handling of such requests should be defined by the policy. The Environment element provides additional context information for the Request and can be used for the Requestor's policy reference, in case of mutual Authorisation.

The AAA Response message format may contain multiple Result elements, as defined by the request message and resource policy. The Result element contains a Decision element, which may contain either "Permit", "Deny" or "Intermediate". The Status element may contain a simple status code (e.g., "OK", "request-info", etc.) and additional status information in the StatusMessage and StatusDetail sub-elements.

The CNLAuthzTicket is generated as the result of a positive PDP decision. It contains the decision and all necessary information to identify the requested service. When presented to the PEP, its validity can be verified and in the case of a positive result, access will be granted without requesting a new PDP decision.

The following describes the current CNLAuthzTicket format and its mapping to the SAML Authorisation Assertion format (due to space limitations , readers are referred to the SAML 1.0 and SAML 2.0 specifications for element names semantics [21, 22]):

- SubjectID and SubjectConfData are placed into SAML Subject/(NameID or BaseID) element and SubjectConfirmation/ConfirmationData element respectively.
- Subject attributes such as JobID and roles are placed into SAMLAuthzDecisionStatement/Evidence element in the form of a SAML Attribute Assertion.
- ValitidyTime containing two attributes "NotBefore" and "NotOneOrAfter" is mapped directly into the related attributes of the SAMLAssertion/Conditions element.
- Other CNLAuthzTicket validity parameters: CommunityRestriction and NumberOfUse – can be placed into related SAML elements or into multiple Conditions/Condition elements.

Another option is to use the XACML profile of SAML 2.0 which allows the inclusion of original XACML Request and Response messages directly into the SAML Authorisation Decision Assertions and Queries [23]. It is our intention to implement this by extending the OpenSAML software libraries to accommodate the new SAML 2.0 specification.

7. CONCLUSION AND SUMMARY

The general OCE Security architecture and proposed solutions described in this paper are based on the practical experience we have gained whilst designing and developing an open collaborative environment within the Collaboratory.nl project. This paper presents findings that resulted from building a flexible, customer-driven security infrastructure for open collaborative applications, based on both the extended use of emerging Web Services and Computer Grid security technologies, and further application specific development of the generic AAA authorisation framework.

The CNL Security Architecture implements the proposed Job-centric approach that allows building basic CNL security services around the semantic Job Description document. The Job Description is created on the basis of signed order- and contains all the information required to run the experiment or execute the job. The Architecture enables security services such as user authentication, policy and role based access control, confidentiality and integrity of information and data.

The CNL Authorisation framework combines Web Services security mechanisms with the flexibility of the Generic AAA Architecture and XACML policy/role based access control model to build fine-grained access control. Separating policy definition from the authorisation enforcement simplifies access control management, which can be delegated to the resource owner. To reduce performance overhead when requesting authorisation decision from PDP, CNL implementation combines pull and push models by using authorisation ticket with the limited validity period that allows bypassing of the potentially slow request evaluation of the PDP.

The CNL project is being developed in coordination with the EGEE project; this will allow future use of the Grid infrastructure being developed in the framework of EGEE project and guarantee the compatibility of basic security services such as authentication, authorisation, and corresponding formats of metadata, policies, messages, etc.

The authors believe that the proposed OCE Security architecture and related technical solutions is relevant for other projects that deal with the development of middleware for virtual laboratories and collaborative applications, especially those which are concerned with secure management of resources in such an OCE.

ACKNOWLEDGEMENT

This paper results from development work conducted within the Collaboratory.nl project, a research initiative that explores the possibilities of remote control and use of advanced lab facilities in a distributed and collaborative industrial research setting. The Collaboratory.nl consortium consists of DSM, Philips, Corus, FEI, Telematica Instituut and the University of Amsterdam.

REFERENCES

[1] "Web Services Architecture". W3C Working Draft 8 August 2003. - http://www.w3.org/TR/ws-arch/

[2] "The Open Grid Services Architecture, Version 1.0. 12 July 2004". - http://www.gridforum.org/Meetings/GGF12/Documents/draft-ggf-ogsa-specv1.pdf

[3] RFC 2903 , Experimental, "Generic AAA Architecture",. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, August 2000 - ftp://ftp.isi.edu/in-notes/rfc2903.txt

[4] RFC 2904 , Informational, "AAA Authorization Framework" J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, D. Spence, August 2000 - ftp://ftp.isi.edu/in-notes/rfc2904.txt

[5] The PRIMA system – Security Mechanisms for Computational Grids. - http://zuni.cs.vt.edu/grid-security/index.html

[6] EGEE Site Access Control Architecture. – DJRA3.2 - https://edms.cern.ch/file/523948/0.20/DJRA3.2-0.20.pdf

[7] Role Based Access Control (RBAC) – NIST, April 2003. - http://csrc.nist.gov/rbac/

[8] eXtensible Access Control Markup Language (XACML) Version 2.0 - Committee Draft 04, 6 December 2004 - http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-04.pdf

[9] GFD.38 Conceptual Grid Authorization Framework and Classification. M. Lorch, B. Cowles, R. Baker, L. Gommans, P. Madsen, A. McNab, L. Ramakrishnan, K. Sankar, D. Skow, M. Thompson - http://www.ggf.org/documents/GWD-I-E/GFD-I.038.pdf

[10] GT 3.9.4 Authorization Framework. - http://www-unix.globus.org/toolkit/docs/development/3.9.4/security/authzframe/

[11] Security Architecture for Open Collaborative Environment, by Yuri Demchenko, Leon Gommans, Cees de Laat, Bas Oudenaarde, Andrew Tokmakoff, Martin Snijders, Rene van Buuren. - European Grid Conference, EGC 2005, Amsterdam, The Netherlands, February 14-16, 2005, Proceedings. Series: Lecture Notes in Computer Science.

[12] Collaboration and security in CNL's virtual laboratory, by Andrew Tokmakoff, Yuri Demchenko and Martin Snijders. WACE 2004, 23 September 2004. -http://www-unix.mcs.anl.gov/fl/flevents/wace/wace2004/talks/tokmakoff.pdf

[13] Demchenko Yu. Virtual Organisations in Computer Grids and Identity Management. – Elsevier Information Security Technical Report - Volume 9, Issue 1, January-March 2004, Pages 59-76.

[14] Web Services Security Framework by OASIS - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[15] Web Services Policy Framework (WS-Policy). Version 1.1. - http://msdn.microsoft.com/ws/2002/12/Policy/

[16] Web Services Policy Attachment (WS-PolicyAttachment). http://msdn.microsoft.com/ws/2004/09/PolicyAttachment/

[17] "A grammar for Policies in a Generic AAA Environment". Work in progress. - http://www.ietf.org/internet-drafts/draft-irtf-aaaarch-generic-policy-06.txt

[18] N. Li, W. Winsborough, J.C. Mitchell. 2003. "Distributed Credential Chain Discovery in Trust Management". *Computer Security.* Vol. 11, No. 1, 2003: 35-86. - http://theory.stanford.edu/people/jcm/papers/disc.pdf

[19] Web Services Agreement Specification (WS-Agreement) - https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecification/en/6

[20] Job Submission Description Language. Version 0.9. - https://forge.gridforum.org/projects/jsdl-wg/document/draft-ggf-jsdl-spec/en/12

[21] Security Assertion Markup Language (SAML) v1.0 - OASIS Standard, Nov. 2002 - http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

[22] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Committee Draft 03,14 December 2004 - http://www.oasis-open.org/committees/download.php/10627/sstc-saml-core-2.0-cd-03.pdf

[23] SAML 2.0 profile of XACML. Draft 02, 11 November 2004. - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf