A Survey on Distributed Access Control Systems for Web Business Processes

Hristo Koshutanski
Department of Computer Science, University of Malaga
Campus de Teations, 29071, Malaga, Spain (Email: hristo@lcc.uma.es)

(Received Nov. 30, 2005; revised and accepted Dec. 31, 2005)

Abstract

Middleware influenced the research community in developing a number of systems for controlling access to distributed resources. Nowadays a new paradigm for lightweight integration of business resources has started to hold Business Processes for Web Services. Authorization and access control policies for Web Services protocols and distributed systems are well-studied and standardized, but there is not yet a comprehensive proposal for distributed access control architecture. This paper surveys the available approaches and analyzes them for a better understanding of what these systems have and what they still need to address the security challenges.

Keywords: Distributed access control, distributed systems security, security architectures, Web services

1 Introduction

Access control has been a constant security issue as the electronic commerce sector has been developed through time. At the end of the past millennium it became an inevitable security issue when the call for *integration of enterprise resources* took a main place in IT development. Middleware was a trendy word connected with products as CORBA, COM+, EJB that emerged at that time. Following this trend a number of access control systems have been developed for distributed e-commerce applications using those technologies. Nowadays a new paradigm for the *lightweight integration of business resources of different enterprises* is starting to take hold – Web Services-based Business Processes. The new paradigm opens new doors of using the available e-commerce systems. Now everything is run over the Web. Web Services are network-accessible using standards as UDDI[1] (discovery), WSDL[2] (interface) and SOAP[3] as a transport protocol that connects them.

The general idea of Web Services (WS for short) is to encapsulate enterprise resources and make them available for use by other enterprises. Moving up in the paradigm, from single partners to orchestration of their business resources, we find virtual enterprises to result. Standards as Business Process Execution Language for Web Services (BPEL4WS) [8] and Electronic Business XML initiative (ebXML)[4] are in place to describe the behavior of complex business and workflow processes.

The basic approaches to distributed authorization, underlying all modern systems and models, are identity-based and capability-based access control. The identity-based approach emphasizes and relies on authentication as a key property for a distributed application. It requires an entity requesting a service to be, first, securely authenticated and then the actual access control decision follows.

Taking access decisions on the basis of requester identity becomes a liability for distributed systems offering their services in an open environment to potentially unknown clients.

Capability-based systems approach distributed authorization in a different and scalable way. Instead of relying on entity's identity they rely on user's capabilities in order to take an access decision. The term *credential* has become widely used for expressing digital access rights in a distributed environment and the *management of credentials* emerged as a key issue for a distributed authorization framework. Thus, credential-based access control [9, 16, 17, 20, 21, 28] becomes the more suitable model for enforcing distributed authorizations.

The notion of credential-based access control has also been referred in the literature as *trust management* [24], especially, in following the early papers by Blaze et al. about KeyNote, PolicyMaker and REFEREE [5, 6, 7, 11]. However, we prefer to use the term credential-based access control. A number of later proposals have refined the languages used for policies, single credentials or hierarchies thereof and for their evaluation [14, 15, 19, 30, 31]. However, the key focus of these proposals is usually the policy and credential language rather than the overall architecture which is responsible for the access decision. Weeks [24] offers a good survey.

[1]www.uddi.org
[2]www.w3.org/TR/wsdl.html
[3]www.w3.org/TR/soap
[4]www.ebxml.org

The overall access control architecture is also important, as we shall in the rest of the paper, that no matter the trust management framework of one's choice there can be many different approaches.

This paper identifies the security requirements for a possible distributed access control system (Section 3). It reviews the major architectural approaches available at industrial and academic environments and summarizes them against the security requirements (Section 4). Thus the paper serves as a good starting point for a better understanding of what the basic components are when one builds a security architecture for Web Business Processes.

2 A Primer on Web Services and Business Processes

A Web Service as defined by the standard [27] is "an interface that describes a collection of operations that are network-accessible through standardized XML messaging. A Web service is described using a standard, formal XML notion, called its *service description*. It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location."

The idea behind Web services is to encapsulate and make available enterprise resources in a new heterogeneous and distributed way.

Web Services Technology Stack

Layer	**Standards**
Workflow	BPEL4WS
Discovery	UDDI
Service Description	WSDL
Messaging	SOAP/XML Protocol
Transport Protocols	HTTP, HTTPS, FTP, SMTP

Figure 1: Web services technology stack

The WS architecture, as defined by W3C[5], is divided into five layers grouped into three main components - Wire, Description and Discovery (Figure 1). The *Wire* component comprises the messaging and transport layers with the SOAP protocol and the XML message format. *Discovery* offers users a unified and systematic way to find, discover and inspect service providers over the Internet. There are two standards proposed at this level - Universal Description, Discovery and Integration (UDDI) and Web Service Inspection Language (WSIL).

[5]W3C Web Services Architecture: http://www.w3.org/TR/ws-arch.

Moving upward we found the *Service Description* layer which is responsible for describing the basic format of offered services (protocols and encodings, where a service resides and how to invoke it). The standard for describing the communication details at this layer is Web Service Description Language (WSDL).

The *Business Process Orchestration* layer is an extension of the service model defined at the description layer. This layer is responsible for describing the behavior of complex business and workflow processes. Intuitively, business processes are graphs where each node represents an orchestration activity and primitive nodes are in WSDL. The proposed standard at this layer is the Business Process Execution Language for WS (BPEL4WS) [8].

The basic BPEL4WS primitive activities are the following:

`<invoke>` invoking an operation on a Web Service.

`<receive>` waiting for an operation to be invoked by someone externally.

`<reply>` generating the response of an input/output operation.

`<assign>` copying data from one place to another.

More complex activities can be constructed by composition:

`<sequence>` allows the developer to define an ordered sequence of steps;

`<switch>` allows the developer to have branching;

`<while>` allows the developer to define a loop;

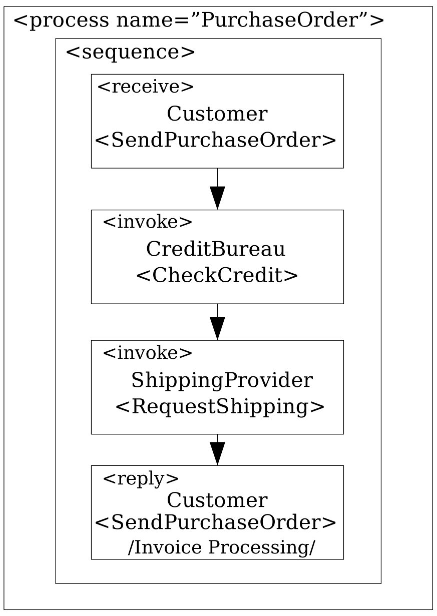
`<flow>` allows the developer to define that a collection of steps has to be executed in parallel.

An example of compositions of services is shown in Figure 2: a buyer service is ordering goods from a seller service, i.e. the buyer service invokes the order method on the seller service, whose interface is defined using WSDL. The seller service invokes a credit validation service to ensure that the buyer can pay for the goods and after that continues by shipping the goods to the buyer. The credit validation service can take place at a credit bureau site in a separate security domain. Notice that a number of partners participate in the process that therefore crosses administrative boundaries.

The XML code shown in Figure 2 is a brief example of the scenario described above in the notations of BPEL4WS primitives. The structure of the processing section is defined by the `<sequence>` element, which states that the elements contained inside are to be executed in this order. The node content is self-explanatory.

3 Authorization Requirements

The advances in communications and networking research brought distributed systems and applications to the forefront place of academic and industrial research. Authorization management has become one of most important issues concerning those systems and applications. The



```
<process name="PurchaseOrder">
    <sequence>
        <receive partner="Customer"
                 portType="purchaseOrderPT"
                 operation="SendPurchaseOrder"
                 container="PO">
        </receive>
        <invoke  partner="CreditBureau"
                 portType="CheckCreditPT"
                 operation="CheckCredit">
        </invoke>
        <invoke partner="shippingProvider"
                portType="shippingPT"
                operation="RequestShipping"
                inputContainer="shipingRequest"
                outputContainer="shippingInfo">
                    <source linkName="ship-to-invoice">
        </invoke>
        <reply partner="Customer"
               portType="purchaseOrderPT"
               operation="SendPurchaseOrder"
               container="Invoice"/>
    </sequence>
</process>
```

Figure 2: Example of a BPEL4WS process

term *distributed authorization* attempts to comprise the whole range of issues from workflow level organizational policies, through service level management of policies, to low-level security mechanisms and architectures for enforcing authorization decisions.

Considering the nature of virtual enterprises – orchestration, choreography, global and local business processes, complex business transactions – the picture changes. Crossing of administrative boundaries becomes the main bottleneck in tailoring the available access control architectures to WS business processes.

Here we identify the security requirements for a possible distributed authorization architecture:

- *Separation of partner's specific security policies and requirements from the authorization system* – a business process spans across many partners, each with its own security policy and requirements. Considering the high degree of autonomy of each partner, it is unrealistic to equalize (restructure) partner's security infrastructure for each inter-organizational workflow.

  Thus an authorization system should stay apart from the internal representation of each partner's policy and how local access decisions are taken. To do so, an authorization system should treat each partner as a distinct object encapsulating its own mechanisms for enforcement of security policy.

- *Support of different policy languages* – this requirement is closely connected with the first one and postulates that a distributed access control system should allow a service provider to define its own security policy in a language best suited for that.

- *Orchestrating requests of grant/deny/additional requirements of many different partners* – in a distributed environment where applications cross several boundaries it becomes difficult and cumbersome to impose a central authority that manages and enforces the requirements and policies of partners from different domains. Moreover some partners may not be willing to disclose their policies directly to the workflow system or even to disclose them at all. So, just combining policies from different application domains is not sufficient.

- *Client/Servent interactive communications* – a client needs a way to fulfill all partners' requirements. Privacy considerations make gathering all potentially needed credentials from a client difficult. Furthermore, this may simply be impossible. An airline may want to ask confidential information directly to its frequent fliers (e.g., confirmation of religious preferences for food) and not to the workflow system. We need a way to interactively disclose required information to the client.

- *Separate entities for policy repository and evaluation* – this simplifies the authorization server's logic and reduces the cost of access control administration.

- *Credential-based access control* – as identified earlier in this section, it is a mechanism that requires a client to provide credentials of access rights (e.g. [28]). These credentials should be acquired before accessing a service and presented at the time of access.

- *Decentralized security administration* – each administrative domain should have full autonomy of specification, management and enforcement of its security policies.

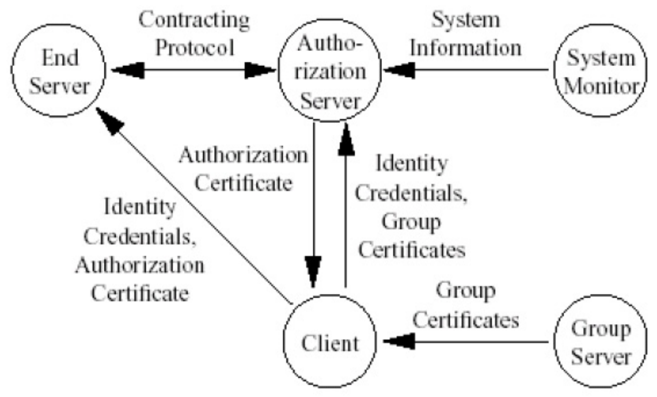


Figure 3: Woo and Lam framework [25]

- *Modular authorization* – allows the authorization service to work with current and future authentication and attribute services.
- *Authorization server as a separate entity* – the main objective is to decouple authorization from application logic. The authorization logic is encapsulated into an authorization service external to the application.

4 Access Control Architectures

If we look at the proposals for distributed access control architectures [2, 4, 10, 12, 13, 23, 25, 29, 32] one of the main design features is splitting the server role into two: an *Application Server* and an *Authorization Server*, i.e. decoupling access control logic from application logic and possibly distribute the access control component [12, 29].

4.1 Single Policy-based Access Control

One of the earliest work on providing a general framework for expressing authorizations was proposed by Woo and Lam [25, 26]. In their work the main component of the system is an Authorization Server that performs authorization on behalf of an End Server. As shown in Figure 3, after a Client has requested the End Server for invoking a service, the End Server elects an Authorization Server in order to offload its access control policy for further evaluation. Then the Authorization Server takes the final access decision and hands out authorization certificates to authorized Clients. These certificates are to be forwarded by the Clients to the End Server along with their requests.

There are three more components in the framework: a System Monitor that tracks the system states; a Group Server that provides group membership information in the form of certificates (membership and nonmembership); and an Authentication Server that authenticates users during their initial sign-on, as well as, performs mutual authentication between every two entities in the system. All components and their message exchanges are shown in Figure 3.

The approach scales well in a distributed environment where each application server can choose its authorization server for getting an authorization decision. The idea of offloading access policies to an authorization server does not fit into the nature of business processes because catering the needs of many different partners from one authorization server makes it complex and heavy in evaluating different authorization policies written in different languages.

Akenti [21] and PERMIS [10] are systems based entirely on digitally-signed documents (certificates). Figure 4 and Figure 5 show Akenti and PERMIS system components, respectively.

Akenti's flow model works as follows. When a client requests an operation it presents an X.509 identity certificate for authentication. The resource server authenticates the client and then asks the Akenti policy engine for an access decision. Akenti checks with the cache server for possibly cached certificates and if that fails, searches certificate directories across the Internet. Once Akenti has all the necessary certificates, it checks whether the client satisfies the requirements to access the resource and returns the access control decision to the resource server. The resource server then enforces the decision and returns the result back to the client. The policy engine either returns "access denied" or a list of actions that are allowed. The resource server must know how to interpret and perform the named actions.

Akenti uses three types of certificates: X.509 user identity certificates for authenticating users, rechange-condition certificates for specifying the conditions that must be met by a user to get access to a resource, and attribute certificates, stored on trusted servers, attesting that a user possesses specific attributes.

Each resource provider (called stakeholder) makes assertions about the conditions that must be satisfied by the user to get access to a resource. These conditions are stated in certificates signed by the stakeholders and located on a web server (local or remote) accessible by the Akenti policy engine. User attributes are asserted and signed by trusted authorities and provided as attribute certificates. So, to face the distributive nature of applications with each resource it is stored a minimal authority file which contains a list of servers that supply the attribute and use-conditions certificates.

The main disadvantage of the approach is that it supports specific (ad-hoc) structure certificates for expressing policies and attributes and the centralized nature of gathering all needed certificates for getting an access decision.

The PERMIS infrastructure consists of two main subsystems: the privilege allocation and the privilege verification subsystem. The former is responsible for assigning privileges to users – issuing X.509 role assignment attribute certificates (ACs), as well as, signing and issuing a service provider's access policy as an X.509 AC. The privilege allocation subsystem stores its ACs in a (local)

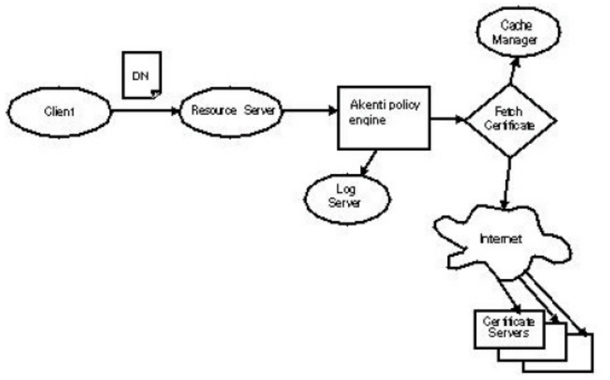


Figure 4: Akenti architecture [21]

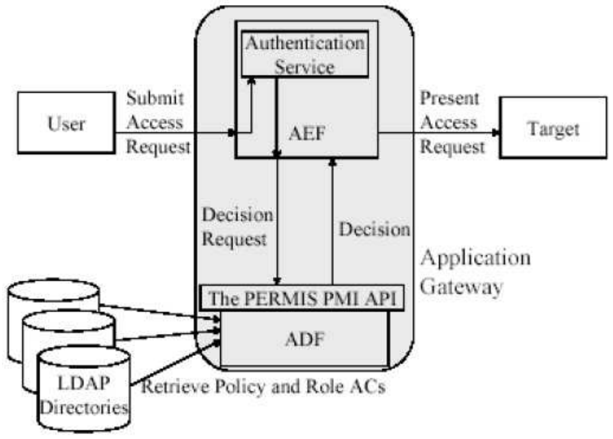


Figure 5: PERMIS architecture [10]

LDAP directory for subsequent use by the privilege verification subsystem.

The privilege verification subsystem (Figure 5) authenticates and authorizes a remote client as well as provides an access decision to target services. In its essence, it is divided in two subfunctions: the Access control Enforcement Function (AEF) and the Access control Decision Function (ADF). The AEF is application dependent. It authenticates the user and enforces the final access decision returned by ADF. The ADF, on the other side, is application independent so it authorizes an already authenticated user in an independent and consistent way.

Another architecture close to PERMIS is the Secure Mediator by Altenschmidt et al. [1]. Again the problem of enforcing authorization is based on digital credentials attesting user's eligibility. The Secure Mediator serves as an integrated view of variety of heterogeneous sources and access to these sources is provided via wrappers. Wrappers can be accomplished by using some distributed object managers (e.g., CORBA, software agents etc). The authors advocate a mediation protocol for secure query answering allowing clients to query resources in a direct manner (with the owner of the resource) or in an indirect way (via the mediator).

The general idea of Akenti, PERMIS and Secure Mediator projects is that the information needed for an access decision, such as identity, authorization, and attributes is stored and conveyed in certificates, which are widely dispersed over the Internet (e.g., LDAP directories, Web servers etc.). The authorization engine has to gather and verify the certificates needed for the user's request and then evaluate them to compute an access decision.

Other two solutions that share common key principles in the design of an authorization service are Adage system [32] (Figure 6) and an architecture [22] (Figure 7) deployed by Hewlett-Packard, called Praesidium authorization server. Both approaches offer centralized security administration and modular authorization. The Application Server communicates with the Authorization Server for obtaining authorization decision. On its side, the authorization server communicates with Identity and Attribute Servers to get additional information for the client. However, here one can spot a sample feature meaningful only for architectures within one administrative domain – during the computation of the access control decision the authorization server determines whether the user needs some roles to be activated and attempts to activate them.

Both systems consist of two domains: administrative domain – concerned with setting up and management of privileges, policies, and profiles granted to principals; and runtime domain – optimized for efficient processing of authorization requests. The latter uses the information available in the administrative domain translated in a form suitable for getting fast and efficient decisions.

Because of the centralized administration of policies and privileges, each partner has to offload (reveal) its own security policies to the authorization server for translation and evaluation.

A widely discussed proposal for distributed access control is OASIS [3]. Figure 8 shows the interactions between a principal and an OASIS secured service. Here, before invoking a service, a principal has to obtain credentials indicating activation of specific roles. To do so, the principal contacts a role activation service, specific per domain, which issues a Role Membership Certificate (RMC) that stores all the credentials the user has activated (Steps 1 & 2 in Figure 8). Role activation is carried out by the Certificate Issuing and Authentication (CIA) service on behalf of all services in a domain. After the certificate is issued, it has to be forwarded by the Client together with a request for a service (Step 3) to the OASIS access control engine. In turn, the access control engine performs a procedure for certificate validation (Step 3 in Figure 8b), by asking an appropriate CIA service, and then enforces access control on the base of client's current credentials and the authorization policy related to the requested service.

The OASIS approach allows decentralized security administration of access control policies for autonomous management domains. It encapsulates each partner's spe-

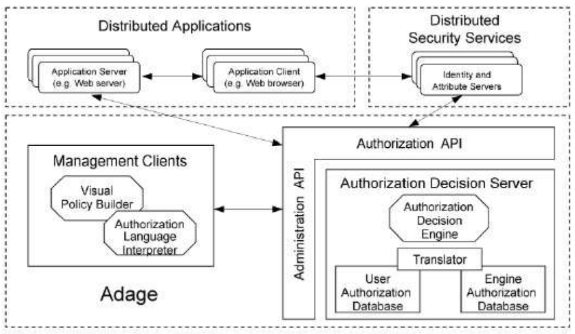

Distributed Applications
Application Server (e.g. Web server)
Application Client (e.g. Web browser)
Distributed Security Services
Identity and Attribute Servers
Authorization API
Management Clients
Visual Policy Builder
Authorization Language Interpreter
Administration API
Authorization Decision Server
Authorization Decision Engine
Translator
User Authorization Database
Engine Authorization Database
Adage

Figure 6: Adage system [32]

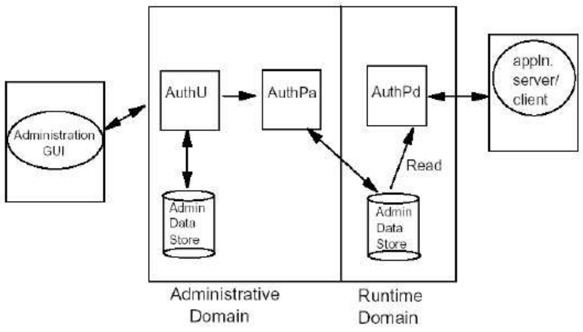


Figure 7: Praesidium authentication server [22]

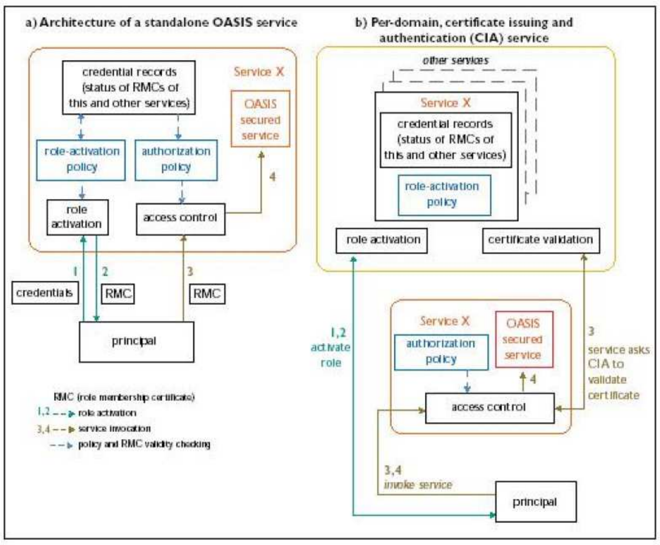

a) Architecture of a standalone OASIS service
b) Per-domain, certificate issuing and authentication (CIA) service
credential records (status of RMCs of this and other services)
Service X
OASIS secured service
role-activation policy
authorization policy
role activation
access control
credentials
RMC
RMC
principal
other services
role activation
certificate validation
1,2 activate role
3 service asks CIA to validate certificate
3,4 invoke service
RMC (role membership certificate)
1,2 role activation
3,4 service invocation
policy and RMC validity checking

Figure 8: OASIS architecture [3]

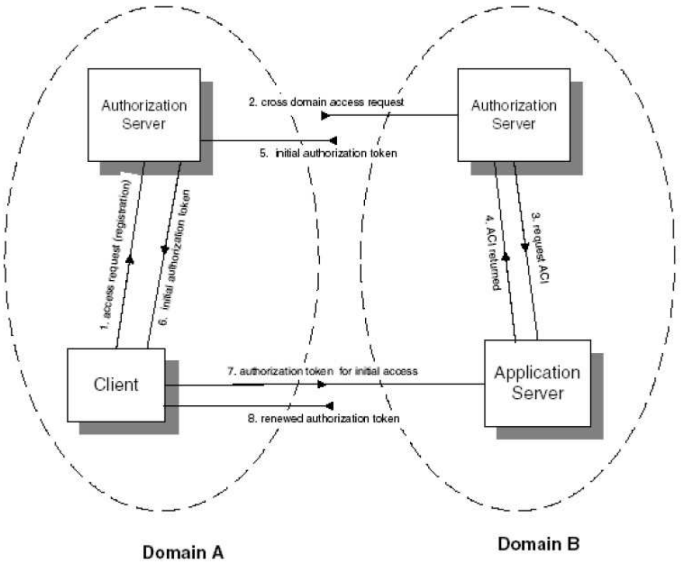

Authorization Server
Authorization Server
2. cross domain access request
5. initial authorization token
1. access request (registration)
6. initial authorization token
3. request ACI
4. ACI returned
Client
Application Server
7. authorization token for initial access
8. renewed authorization token
Domain A
Domain B

Figure 9: Cross-domain authorization [2]

cific policy with its internal interpretation and evaluation and interoperates requirements for credentials to service level agreements.

The work by Au et al. [2] proposes a technique of using one-shot authorization tokens (Figure 9). A smart card is adopted as an authorization device that stores client's tokens in a secure and mobile way. In the proposed authorization scheme there are three main elements: an authorization server; a client workstation; and an application server. The role of the authorization server is to provide initial credentials in the form of an authorization token to all users under the same administrative domain and to administer centrally the access control information at each application server. It is also responsible for the communications with other authorization servers from different domains in order to set up the user's initial access rights. Thus, each partner does not need to offload (reveal) its policy to the partner orchestrator of the process, but just to issue credentials in a secure token.

4.2 Multi-Policy Based Access Control

A step closer to orchestrating Web Services is by Beznosov et al. [4]. In this work authorizations are managed by an Authorization Service and its Access Decision Object (ADO). Figure 10 shows a message flow between entities in the architecture for computing an access decision.

The main advantage of the approach is the use of Policy Evaluators. Policy Evaluators serve as distinct authorities each with its own security policies. Their role is to encapsulate different authorization policies with their internal representation and evaluation – a step ahead for addressing the security requirement for separation of partner's specific security infrastructure from the authorization system.

The sequence of messages leading to an access decision is the following. An application server contacts ADO server for an authorization decision. ADO obtains references to all policy evaluators related to the client's request, asks a decision combinator for combining decisions returned by the various evaluators (according to a suit-

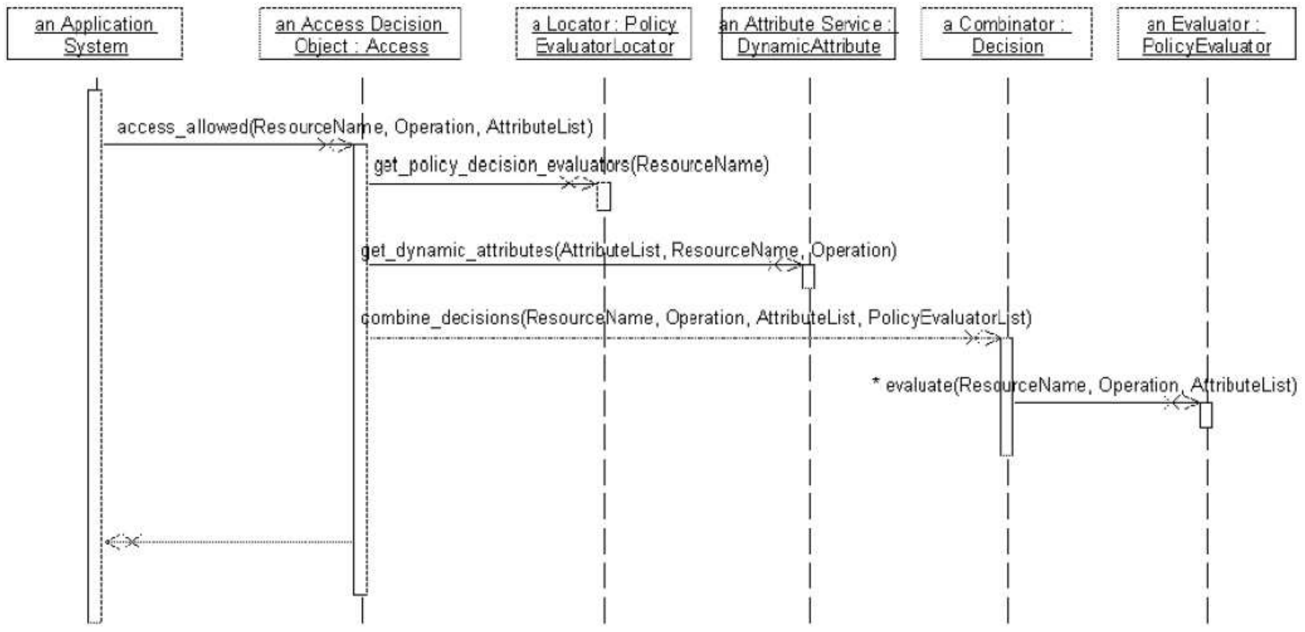

an Application System
an Access Decision Object : Access
a Locator : Policy EvaluatorLocator
an Attribute Service : DynamicAttribute
a Combinator : Decision
an Evaluator : PolicyEvaluator
access_allowed(ResourceName, Operation, AttributeList)
get_policy_decision_evaluators(ResourceName)
get_dynamic_attributes(AttributeList, ResourceName, Operation)
combine_decisions(ResourceName, Operation, AttributeList, PolicyEvaluatorList)
* evaluate(ResourceName, Operation, AttributeList)

Figure 10: RAD architecture [4]

able combination policy), and returns the decision back to the application server (see Figure 10).

The Policy Evaluator returns to the Decision Combinator, as a result of the policy evaluation, reply *yes/no/don't know* decision. This is a step towards addressing the requirement of orchestrating partners' authorization requests. It's a possible approach for orchestrating authorization requests although it is not fully on target.

A more advanced effort for enforcing and administrating authorization policies across heterogeneous systems is the OASIS eXtensible Access Control Markup Language (XACML) framework [29]. The main actor here is the Policy Decision Point (PDP) responsible for retrieving the relevant policies with respect to the client's request, evaluating them and rendering an authorization decision. The work also considers the combination of different policies from various partners using some policy combining algorithms and getting an authorization decision on the base of evaluating them. In this case PDP has access to a partner's security policy, i.e. every partner in a process has to reveal its security policy to the PDP in order to be computed an access decision. However, we need a way to orchestrate different partners' access decisions (requests) instead of just combining their policies.

The other key approach of OASIS consortium is the Secure Assertion Markup Language (SAML) standard [18]. The main objective of the approach is to offer a standard way for exchanging authentication and authorization information between trust domains. The basic data objects of SAML are assertions. Assertions contain information that determines whether users can be authenticated or authorized to use resources. The SAML framework also defines a protocol for requesting assertions and responding to them, which makes it suitable when modeling interactive communications between entities in a distributed environment.

4.3 Access Control Systems and Security Requirements

Table 1 summarizes the access control systems described above and compares them against the security requirements stated in Section 3.

5 Conclusions

There are many access control models for Web Services and XML documents. For virtual enterprises the picture changes. WS business processes describe the behavior of complex business logic and form the so called Web services workflow. Web services workflow may contain many tasks of different levels of abstraction: a (recursive) reference to another Web services workflow; a simple task performed by a computer program, a database transaction, etc. On these levels of abstraction applying the already reviewed architectural approaches for access control is no longer sufficient.

Looking at Table 1 we find a good approximation over the authorization requirements in proposals [4, 3, 29, 18].

However, orchestrating partners' (end-point) access decisions on the workflow level, the major bottleneck for web business processes, is not well captured and is still an open issue.

Because of the pervasive nature of integrating business resources from different application domains there is a

Table 1: Summary of the access control systems and the authorization requirements

AC Systems \ Features	Auth. Server	Modular Auth.	Decentralized Security Admin.	Credential/ Certificate based AC	Separate entities for policy repository and evaluation	Support different auth. languages	Combining different sec. policies	Orchest. partners' auth. requests	Client/ Servent interactive comm-ns	Separation partners' spec. sec. policies from AC system
Woo&Lam	√	√	√	√	√					
Akenti	√		√	√	√					
PERMIS	√	√	√	√	√					
RAD	√	√	√		√	√	√	√		√
Adage	√	√								
Praesidium	√	√								
OASIS	√	√	√	√	√	√				√
One-shot auth. token	√	√		√	√					√
OASIS (XACML&SAML)	√	√	√	√	√		√		√	

pressing need for a proposal that synthesizes all the above mentioned aspects into one access control architecture for Business Processes for Web Services.

Acknowledgements

This work was partially supported by the projects: 016004 IST-FP6-FET-IP-SENSORIA, 2003-S116-00018 PAT-MOSTRO and FIRB programme of MIUR under the RBNE0195K5 ASTRO.

References

[1] C. Altenschmidt, J. Biskup, U. Flegel, and Y. Karabulut, "Secure mediation: Requirements, design, and architecture," *Journal of Computer Security*, vol. 11, no. 3, pp. 365-398, 2003.

[2] R. Au, M. Looi, and P. Ashley, "Cross-domain one-shot authorization using smart cards," in *Proceedings of the 7th ACM conference on Computer and communications security*, pp. 220-227, ACM Press, 2000.

[3] J. Bacon and K. Moody, "Toward open, secure, widely distributed services," *Communications of the ACM*, vol. 45, no. 6, pp. 59-64, 2002.

[4] K. Beznosov, Y. Deng, B. Blakley, C. Burt, and J. Barkley, "A resource access decision service for CORBA-based distributed systems," in *Proceedings of 15th IEEE Annual Computer Security Applications Conference (ACSAC'99)*, pp. 310–319, 1999.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, "The role of trust management in distributed systems security," in *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, pp. 185-210, Springer-Verlag, 1999.

[6] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "KeyNote: Trust management for public-key infrastructures," in *Proceedings of 6th International Workshop on Security Protocols*, LNCS 1550, pp. 59-63, Springer-Verlag, 1998.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 164-173, 1996.

[8] BPEL4WS, *Business Process Execution Language for Web Services (BPEL4WS)*, 2003 (http://www-106.ibm.com/developerworks/webservices/library/ws-bpel).

[9] D. Chadwick, A. Otenko, and E. Ball, "Role-based access control with X.509 attribute certificates," *IEEE Internet Computing*, vol. 7, no. 2, pp. 62-69, Mar/Apr. 2003.

[10] D. W. Chadwick and A. Otenko, "The PERMIS X.509 role-based privilege management infrastructure," in *Seventh ACM Symposium on Access Control Models and Technologies*, pp. 135-140, 2002.

[11] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "REFEREE: Trust management for Web applications," *Computer Networks and ISDN Systems*, vol. 29, pp. 953-964, no. 8-13, 1997.

[12] J. A. Hine, W. Yao, J. Bacon, and K. Moody, "An architecture for distributed OASIS services," in *IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 104–120, Springer-Verlag, 2000.

[13] W. Johnston, S. Mudumbai, and M. Thompson, "Authorization and attribute certificates for widely distributed access control," in *Proceedings of Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98)*, pp. 340–345, 1998.

[14] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 119-130, 2002.

[15] N. Li, W. H. Winsborough, and J. C. Mitchell, "Distributed credential chain discovery in trust management," *Journal of Computer Security*, vol. 11, no. 1, pp. 35-86, Feb. 2003.

[16] R. Oppliger, A. Greulich, and P. Trachsel, "A distributed certificate management system (DCMS) supporting group-based access controls," in *Proceedings of 15th IEEE Annual omputer Security Applications Conference (ACSAC'99)*, pp. 241-248, 1999.

[17] J. S. Park and R. Sandhu, "RBAC on the Web by smart certificates," in *Proceedings of the Fourth ACM Workshop on Role-based Access Control*, pp. 1–9, 1999.

[18] SAML, *Security Assertion Markup Language (SAML)*, 2004 (http://www.oasis-open.org/committees/security).

[19] K. Seamons and W. Winsborough, *Automated Trust Negotiation*, Technical report, US Patent and Trademark Office, 2002. IBM Corporation, patent application filed Mar. 7, 2000.

[20] SPKI, *SPKI Certificate Theory*, IETF RFC 2693, 1999.

[21] M. Thompson, W, Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari, "Certificate-based access control for widely distributed resources," in *Proceedings of Eighth USENIX Security Symposium (Security'99)*, pp. 215-228, Aug. 1999.

[22] V. Varadharajan, C. Crall, and J. Pato, "Authorization in enterprise-wide distributed system: a practical design and application," in *Proceedings of 14th IEEE Annual Computer Security Applications Conference*, pp. 178–189, 1998.

[23] V. Varadharajan, C. Crall, and J. Pato, "Issues in the design of secure authorization service for distributed applications," in *Global Telecommunications Conference (GLOBECOM'98) - The Bridge to Global Integration*, vol. 2, pp. 874-879, IEEE Press, 1998.

[24] S. Weeks, "Understanding trust management systems," in *IEEE Symposium on Security and Privacy (SS&P'01)*, pp. 94-105, 2001.

[25] T. Y. C. Woo and S. S. Lam, "Designing a distributed authorization service," in *Proceedings of Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'98)*, vol. 2, pp. 419–429, 1998.

[26] T. Y. C. Woo and S. S. Lam, "A framework for distributed authorization", in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 112-118, 1993.

[27] WS ConceptualArchitecture, *Web Services Conceptual Architecture (WSCA 1.0)*, May 2001 (http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf).

[28] X.509, *The directory: Public-key and attribute certificate frameworks*, ITU-T Recommendation X.509:2000(E) | ISO/IEC 9594-8:2001(E), 2001.

[29] XACML, *eXtensible Access Control Markup Language (XACML)*, 2004 (http://www.oasis-open.org/committees/xacml).

[30] W. Yao, "Fidelis: A policy-driven trust management framework," in *iTrust*, LNCS 2692, pp. 301-314, Springer-Verlag, 2003.

[31] W. Yao, *Trust management for widely distributed systems*, PhD thesis, University of Cambridge, Computer Laboratory, 15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom, 2004. Appeared as a technical report UCAM-CL-TR-608, ISSN 1476-2986.

[32] M. Zurko, R. Simon, and T. Sanfilippo, "A user-centered, modular authorization service built on an RBAC foundation," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 57-71, 1999.

Hristo Koshutanski received M.Sc. in Mathematics from Plovdiv University "Paisii Hilendarski" in 2001 and Ph.D. in Information and Communication Technology from University of Trento in 2005. He has been a post doc researcher in the field of security and trust at CREATE-NET center during Sept 2005 - Sept 2006. He is currently a post doc researcher in computer science department at the University of Malaga.

In 2005 he won the E-NEXT SATIN award (The European Doctoral School of Advanced Topics In Networking) for doctoral research and in August 2005 he was a lecturer at ESSLLI'05 European summer school.

Dr. Koshutanski won EU Marie Curie EIF Fellowship with host institution University of Malaga for the period March 2007- March 2009. His research interests include distributed access control models, trust management techniques for digital credential negotiation, semantics of access control, digital identity management. He has co-authored a number of scientific papers. Contact him at: hristo@lcc.uma.es