UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

DIS TRIBUTED ACCESS CONTROL FOR WEB AND BUSINESS PROCESSES

Hristo Koshutanski

June 2003

Technical Report # DIT-03-034

Also: submitted to IEEE Internet Computing (January/February 2004)

Distributed Access Control for Web and Business Processes

A Survey at the Beginning of the Millennium

Hristo Koshutanski
Dip. di Informatica e Telecomunicazioni - Univ. di Trento
via Sommarive 14 - 38050 Povo di Trento (ITALY)
hristo.koshutanski@dit.unitn.it

June 2003

Abstract

Middleware influenced the research community in developing a number of systems for controlling access to distributed resources. Nowadays a new paradigm for the lightweight integration of business resources from different partners is starting to take hold – Web Services and Business Processes for Web Services.

Security and access control policies for Web Services protocols and distributed systems are well studied and almost standardized, but there is not yet a comprehensive proposal for an access control architecture for business processes. So, it is worth looking at the available approaches to distributed authorization as a starting point for a better understanding of what they already have and what they still need to address the security challenges for business processes.

Keywords: survey, distributed access control, security architecture, distributed systems security, web services, web services processes, web services security.

Contents

1 Connection on Web and Business Processes

Access control has been a constant security issue as the IT sector has been developed through the 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. Nowadays a new paradigm for the *lightweight integration of business resources of different enterprises* is taking place – Web Services and Business Processes for Web Services. Now everything is run over the Web. Web Services are network-accessible using standards as UDDI (discovery), WSDL (interface) and SOAP 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 using by other enterprises. Moving up in the paradigm from single enterprises to orchestration of their business resources we find virtual enterprises to result. Here the proposed standards as Business Process Execution Language for Web Services (BPEL4WS), Electronic Business XML initiative (ebXML) are in place to describe the behavior of complex business and workflow processes. For a primer on Web Services and business processes see Appendix.

Since middleware became a main paradigm for offering a transparent view of distributed resources of a single enterprise much work has been done in controlling access to those resources. We find on the research market a number of papers for access control models for basic WS and XML documents [9, 14, 3] and architectural approaches (Section 3) for enforcing access control models in a distributed enterprise environment.

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

The goal of this paper is to survey the architectural approaches for distributed access control available at academic and industrial environments. It is a good starting point for a better understanding of what are the basic components one needs in building a security architecture for business processes.

The remainder of this paper is organized as follows. Section 2 explains the security requirements for an access control architecture for WS business processes. Then the architectural overview of the systems for distributed authorization is presented in Section 3. Section 4 summarizes the described access control systems and the security features available at each of them and concludes the paper.

2 Security Requirements for Web and Business Processes

This section describes the security requirements that have to be considered in building an access control architecture for business processes.

Let us consider again the nature of virtual enterprises - many partners each with its own security policy and requirements. It is unrealistic to equalize (restructure) each partner's security infrastructure for each inter-organizational workflow. This identifies the first security requirement - *separation of partners' specific security policies and requirements from the WS workflow.*

The next step in considering security requirements is how to express security policies at workflow level. Different partners with different security policies are just partners in the Web process, so they may not be willing to disclose their policies directly to the workflow system or even may not want to disclose them at all, so just combining them is not sufficient. This identifies the second security requirement - *orchestrating requests grant/deny/counter request (additional requirements) of many different partners.*

As mentioned before because of the heterogeneous nature of WS processes a client needs a way of fulfilling 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., conformation of religious preferences for the food) and not to the workflow system. This calls for the third security requirement - *client-servent*[1] *interactive communications.* We need a way of how to incrementally disclose required information to the client.

The following security requirements are implications from the above discussed ones:

- Support of different authorization languages – in order to separate partners' specific security policies and requirements from the access control system, we should abstract from the internal representation of each partner's policies and how the local access decision is taken. So, an access control system should treat each partner as a distinct object encapsulating its own authorization policy (written in a specific language) with an engine for an appropriate evaluation. The partner's policy evaluator should be accessible by the access control system in a unified way;

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

- Credential/Certificate based access control – it is a mechanism that requires the client to provide a certificate (a ticket) that indicates some type of access right. The mechanism also requires the client to acquire this certificate before performing an access and presenting it at the time of access. This mechanism prevails the classical role-based access control because all the information necessary for

[1] Here the servent is considered as an entity that acts either as a server or as a client.

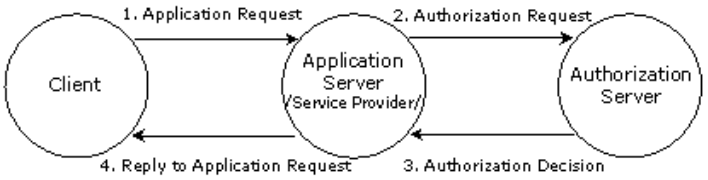


Figure 1: Splitting the server role

computing an access decision is available outside of the access control system - in the partners local policies. While in the classical role-based access control all the information needed for computing client's access right is internally available in the system. So, the client just needs to authenticate itself to the system;

- Decentralized security administration – because of the distributed nature of WS processes administration of security policies should be done autonomously and independently by the partners' specific system administrators. Coordinating policy across different domains is not straightforward;
- Modular authorization – it allows the authorization service to work with current and future authentication and attribute services, i.e. the authorization service depends on the authentication service for reliable authentication of principals, but should not depend on the specific kind of authentication mechanism;
- Authorization server as a separate entity – the main objective of this separation is to decouple authorization logic from application logic. The authorization logic is encapsulated into an authorization service external to the application. This approach significantly reduces the cost of access control administration, as well as, enables coherent and consistent authorization policies across heterogeneous systems.

3 Distributed Authorization Architectures

Because web service arena is highly distributed and heterogeneous, and is open to the public world there is a pressing need to control *who can access what kind of resources under what conditions*. And the most important question now is how to enforce and administrate a security policy across an entire environment with multiple distributed and heterogeneous systems.

The goal of this section is to survey the architectural approaches and solutions, attempting to solve the above mention problem, available at academic or industrial environment.

If we look at the proposals for distributed access control architectures [21, 13, 5, 23, 1, 19, 11, 7, 10] the main building block is splitting the server role into two: an *Application Server* and an *Authorization Server*, i.e. decoupling access control logic from application logic (see Figure 1), and possibly distribute the access control component [11, 10].

One of the earliest work on providing a general framework for expressing authorizations was proposed by Woo and Lam [22, 21]. In their work the main component of the system is an Authorization Server

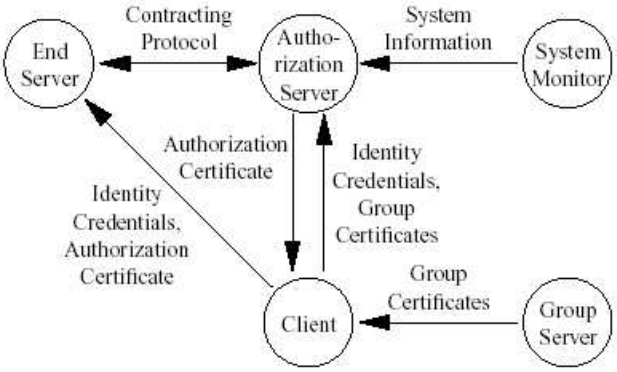


Figure 2: Message exchanges in Woo and Lam framework [21, page 3]

that performs authorization on behalf of an End Server. As shown in Figure 2, 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 tree 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 the components and their message exchanges are shown in Figure 2.

The approach scales well in the distributed WS environment where each basic Web service can choose its authorization server for getting an authorization decision. But the idea of "offloading access policies" does not fit into the nature business processes, because catering the needs of many different partners from one authorization server (as the idea is) makes it complex and heavy in evaluating different authorization policies written in different languages.

Other two approaches, which are based entirely on digitally-signed documents (certificates), are Akenti [17, 13] and PERMIS [7]. The general idea of the projects is that all the information needed for getting 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.). Both approaches use three types of certificates: X.509 identity certificates for authenticating the users; certificates for storing access control policies; and certificates for storing users' credentials. Figures 3 and 4 show the major components of Akenti and PERMIS systems and how these components communicate each other for geting an authorization decision. In general the authorization engine has to gather and verify all the certificates needed for the current user's request and evaluate the user's right to access it based on these certificates. Evaluating the certificates is consistent because are generated by standard tools available at each system.

Considering the above two systems from the point of view of Web and business processes one can

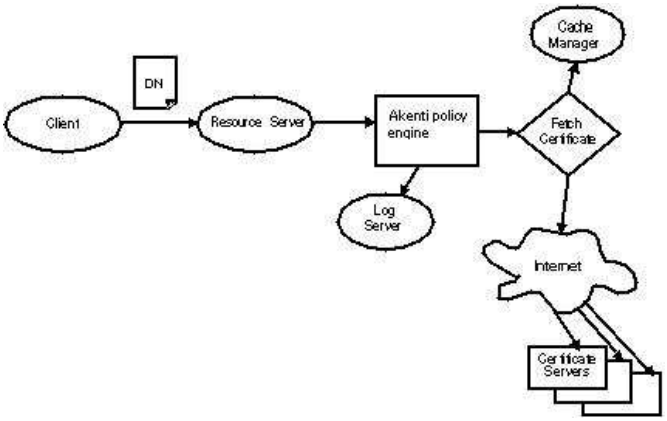


Figure 3: Overview of Akenti Architecture [17, page 3]

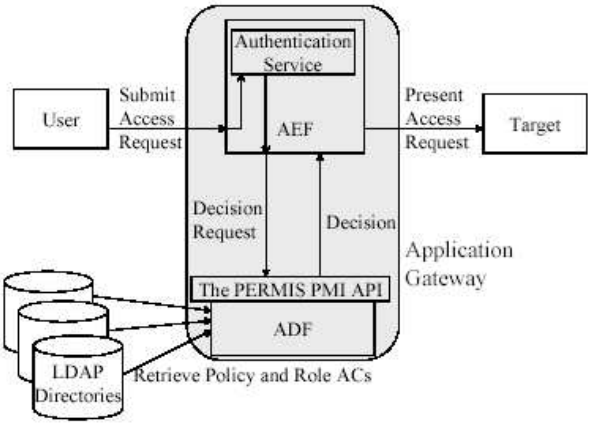


Figure 4: PERMIS Architecture [7, page 5]

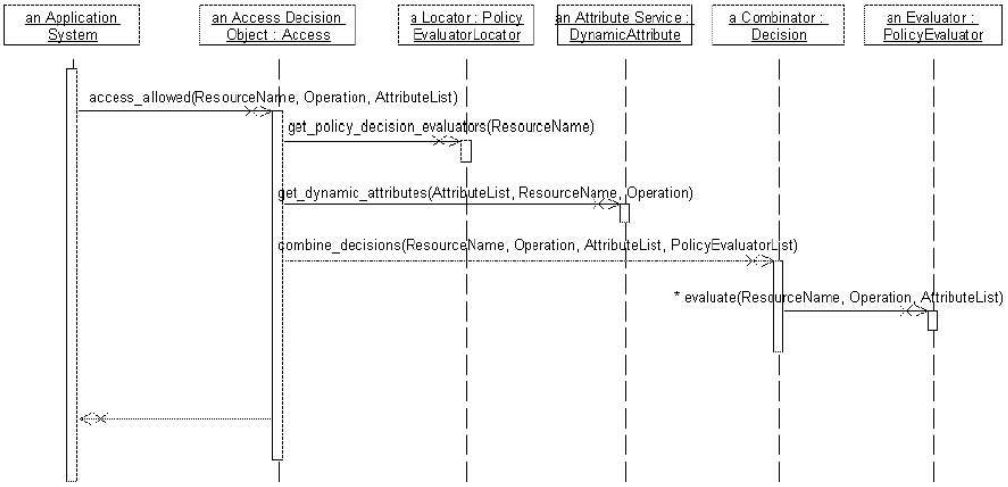


Figure 5: RAD Architecture [5, page 7]

use them (with modifications) for achieving access control for business processes in a very rough way – collecting all needed partners' policies (contained in certificates) related to the current process and evaluating them. Modifications are needed to specify how to combine different partners' policies.

A step closer to what we need for orchestrating Web services is in [5]. In the work authorizations are managed by an Authorization Service and its Access Decision Object (ADO). The ADO obtains references to all policy evaluators related to the client request, asks a decision combinator for combining decisions returned by various evaluators (according to a suitable combination policy), and returns the decision back to the client (see Figure 5).

The proposal uses Policy Evaluators 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 closer to the security requirement for separation of partner's specific security infrastructure from WS workflow.

Other solutions that have common key principles in the design of an authorization service for dis-

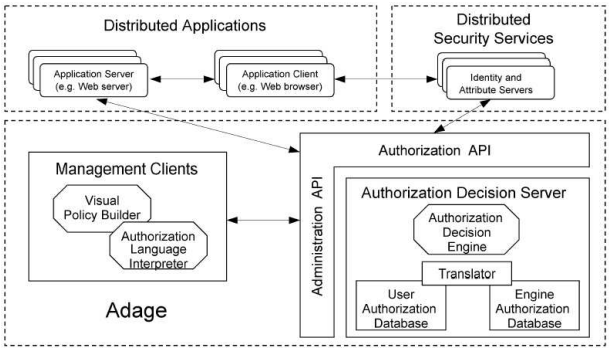


Figure 6: Adage Architecture [23, page 4]

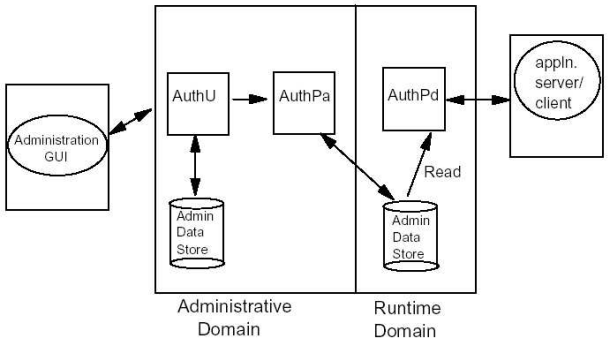


Figure 7: Praesidium Authorization Server [18, page 5]

tributed applications can be found in [23, 18]. In [23] is presented the architecture of *Adage* system (Figure 6) and in [18] is proposed the architecture of an authorization server that has been deployed by Hewlett-Packard and called *Praesidium* authorization server (Figure 7). Both approaches offer:

- centralized security administration - for establishing security policies clients have to communicate with the authorization server through administrative API. Here clients are categorized by the interface(s) they access (management clients or application clients)
- modular authorization - it allows the authorization service to work with current and future authentication and attribute services.

Again here the Application Server communicates with the Authorization Server for obtaining authorization decision (see Figures 6 and 7). The authorization server by its side communicates with Identity and Attribute Servers in order to obtain additional information for the client. However – here one can spot a sample feature making sense only for architectures working 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 tries to activate them on its own.

Each of the two systems can be considered as consisting of two domains: Administrative domain – concerned with the setting up and management of privileges, policies, and profiles granted to principals. Runtime domain – optimized for efficient processing of authorization requests. It uses the information available at administrative domain translated in a form suitable for getting fast and efficient decisions.

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

If we continue looking for other real-world approaches, we will inevitably encounter the OASIS architecture [2, 11]. Figure 8 shows the interactions between a principle and an OASIS secured service. In these interactions the first thing (steps 1,2 in Figure 8a) that the principal has to do in order to invoke a service is to obtain enough credentials, i.e. to activate some (specific) roles. To do this it needs to contact a role activation service, specific per domain-based, which as a result issues a Role Membership Certificate (RMC) that stores all the credentials the user has activated. Role activation is carried out by the Certificate Issuing and Authentication (CIA) service on behalf of all services in a domain. After the certificate has been issued it has to be forwarded by the Client together with a request for the service

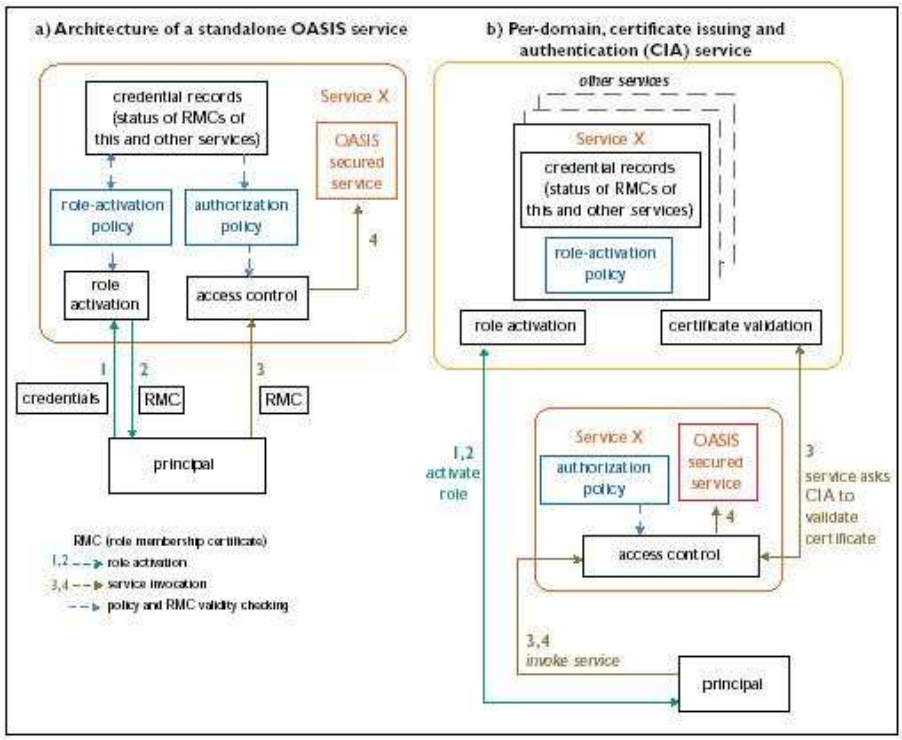


Figure 8: OASIS Architecture [2, page 3]

(step 3 in Figure 8) to the OASIS access control engine. By its side the access control engine performs a procedure for certificate validation (step 3 in Figure 8b), in 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. If the result from the evaluation is positive then is invoked the requested service and the result is returned back to the client (step 4 in Figure 8).

The proposal well encapsulates each partner's (domain) requirements for credentials with their internal interpretation and evaluation. So, in this case a partner does not need to reveal its security requirements to another server orchestrator of a process, but present it as a service. Trying to fit this into the WS business processes picture one meets difficulties in orchestrating independent partners' access control policies.

Another approach for distributed authorization can be found in [1]. In this paper is proposed a technique of using one-shot authorization tokens. A smart card is used 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: a security server; a client workstation; and an application server. The security server provides centralized security services such as authentication and authorization, as well as, administers all the application servers and users in the same security domain. The authorization server administers centrally the access rights of all authenticated users – in providing initial access rights (credentials) to them in the form of authorization token; and administers the access control information in each application server - in updating and revoking users' tokens in the application servers' ACLs.

Considering again the nature of Web business processes here the authorization server communicates with other authorization servers from different domains (partners) in order to set up credentials and requirements in the form of a token (see Figure 9). In this case each partner does not need to offload (reveal) its policy to another partner orchestrator of the process, but just issues credentials and/or

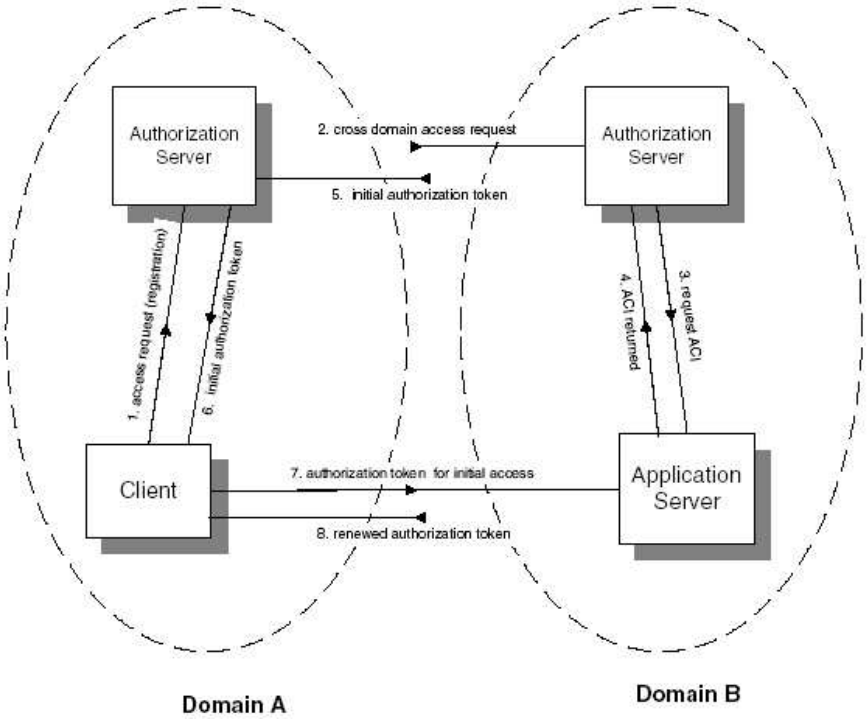


Figure 9: Cross-domain Authorization [1, page 6]

requirements as a secure token. Again here arises the problem of orchestrating different partner's decisions and presenting them to the client.

A more advanced effort for enforcing and administrating authorization policies across heterogeneous systems is the OASIS security framework [16, 10]. The main actor in the framework is the Policy Decision Point (PDP) responsible for retrieving the applicable to the client's request policies, evaluating them, and rendering an authorization decision (see Figure 10). The work also considers the combination of different policies from different partners using some policy combining algorithms and getting an authorization decision on the base of evaluating them. In this case the PDP has access to each partner's security policy, i.e. each partner has to reveal its security policy to the PDP in order to be computed an access decision on the overall business process. But as we have specified in Section 2 we need a way of how to orchestrate different partners' security policies and requirements instead of just combining them.

4 Conclusions

As we have have hinted at the beginning of the paper there are many access control models for Web services and XML documents [9, 14, 3, 10, 8]. Stepping towards virtual enterprises the picture changes. WS business processes describe the behavior of complex business logic and in this way forming the so called Web services workflow. Web services workflow may contain many tasks of different levels of abstraction where a task may be a (recursive) reference to another Web services workflow or a simple one performed by a computer program, database transaction, etc. Considering these levels of abstraction applying the already reviewed architectural approaches for access control is no longer applicable (sufficient).

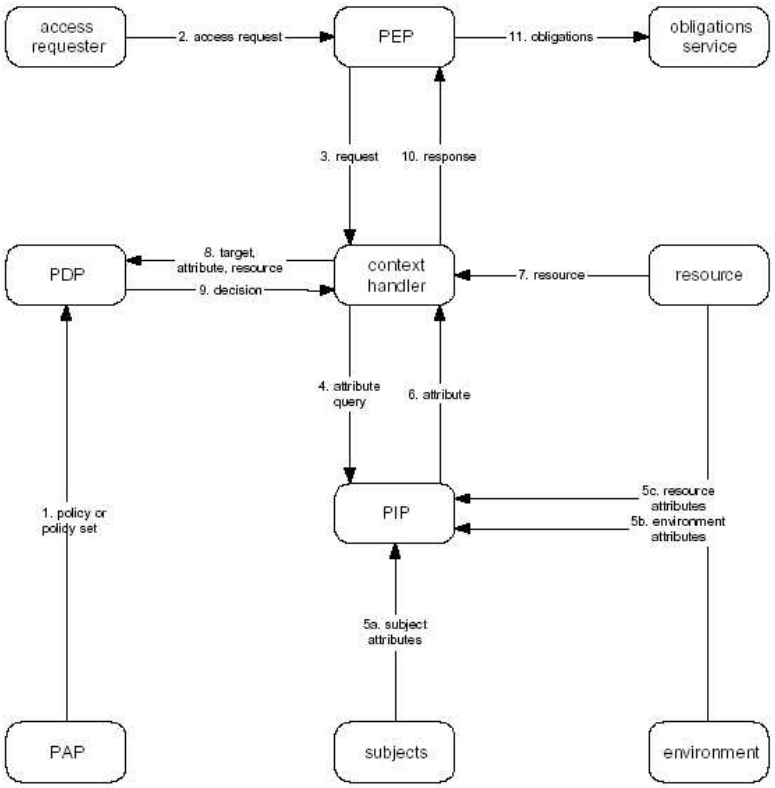


Figure 10: XACML data-flow diagram [10, page 19]

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	Interactive Client/ Servent 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)	√	√	√	√	√	×	√	×	×	×
Web&Business Processes	√	√	√	√	√	√	√	√	√	√

Table 1: Access control systems and security features

Table 1 summarizes the access control systems described in the previous section and compares each of them against the security requirements specified in Section 2. The last row of Table 1 identifies the security requirements for a possible Web and business process access control system.

Looking at the first security requirement (Section 2) and considering Table 1 we find a good approximation in proposals [5, 2, 1] in which we need a unified way of asking different partners involved in a business process for their access decisions – grant/deny/counter request (additional requirements).

Looking at the second security requirement (Section 2) we find at the access control field a good approximation of combining policies at the logical level [4, 15, 20, 12] and (considering Table 1) at the

architectural level [10, 5].

In most proposals, the possibility that servers may get back to the calling clients with some counter requests is not considered. This even in the case where the client is actually an authorization server querying different partners' policy evaluators – the third security requirement. Here we find a proposal [6] for controlling release of information. The work proposed by Bonatti and Samarati introduces the idea of interactive release of information in a single client/server communications. It proposes a framework allowing two parties to communicate their requirements with respect to the information disclosure rules and a language for the specification of such requirements. This work can be used as an endpoint access control model (on the level of basic Web services – see Figure 11 in Appendix) in a possible architecture for business processes .

Because of the dynamic and pervasive nature of business processes there is a pressing need for a proposal that synthesizes all the above mentioned aspects into one access control architecture for business processes for Web services – an area that should be investigated in the near future.

5 BOX: Web Services and Business Processes on the Web

Web Services Miscellaneous – www.webservices.org

WSDL 1.1 specification – www.w3.org/TR/wsdl.html

UDDI pages – www.uddi.org

BPEL4WS 1.0 specification – www-106.ibm.com/developerworks/ webservices/library/ws-bpel/

WSIL 1.0 specification – www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html

W3C Web Services Architecture – www.w3.org/TR/ws-arch

Web Services Conceptual Architecture (WSCA 1.0) – http://www-3.ibm.com/software/solutions /web-services/pdf/WSCA.pdf

ebXML Business Process Specification Schema v1.01 – www.ebxml.org/specs/ebBPSS.pdf

References

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

[2] Bacon, J., and Moody, K. Toward open, secure, widely distributed services. *Communications of the ACM 45*, 6 (2002), 59–64.

[3] Bertino, E., Castano, S., and Ferrari, E. On specifying security policies for Web documents with an XML-based language. In *Proceedings of the Sixth ACM Symposium on Access control models and technologies* (2001), ACM Press, pp. 57–65.

[4] Bertino, E., Ferrari, E., and Atluri, V. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC) 2*, 1 (1999), 65–104.

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

[6] Bonatti, P., and Samarati, P. A unified framework for regulating access and information release on the Web. *Journal of Computer Security*. (to appear).

[7] Chadwick, D. W., and Otenko, A. The PERMIS X.509 role-based privilege management infrastructure. In *Seventh ACM Symposium on Access Control Models and Technologies* (2002), ACM Press, pp. 135–140.

[8] Damiani, E., di Vimercati, S. D. C., Paraboschi, S., and Samarati, P. Fine grained access control for SOAP E-services. In *Proceedings of the tenth international conference on World Wide Web* (2001), ACM Press, pp. 504–513.

[9] Damiani, E., Samarati, P., De Capitani di Vimercati, S., and Paraboschi, S. Controlling access to XML documents. *IEEE Internet Computing 5*, 6 (Nov.-Dec. 2001), 18–28.

[10] Godik, S., and Moses, T. *eXtensible Access Control Markup Language (XACML)*. OASIS, February 2003. www.oasis-open.org/committees/xacml/.

[11] Hine, J. A., Yao, W., Bacon, J., and Moody, K. An architecture for distributed OASIS services. In *IFIP/ACM International Conference on Distributed systems platforms* (2000), Springer-Verlag New York, Inc., pp. 104–120.

[12] Jajodia, S., Samarati, P., Subrahmanian, V. S., and Bertino, E. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data* (1997), ACM Press, pp. 474–485.

[13] Johnston, W., Mudumbai, S., and Thompson, M. 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)* (1998), IEEE Press, pp. 340–345.

[14] Joshi, J. B. D., Aref, W. G., Ghafoor, A., and Spafford, E. H. Security models for web-based applications. *Communications of the ACM 44*, 2 (2001), 38–44.

[15] Li, N., Grosof, B. N., and Feigenbaum, J. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC) 6*, 1 (2003), 128–171.

[16] OASIS Security Services TC. *Security Assertion Markup Language (SAML)*. OASIS, November 2002. www.oasis-open.org/committees/security/.

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

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

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

[20] Wijesekera, D., and Jajodia, S. Policy algebras for access control the predicate case. In *Proceedings of the 9th ACM conference on Computer and Communications Security* (2002), ACM Press, pp. 171–180.

[21] Woo, T. Y. C., and Lam, S. Designing a distributed authorization service. In *Proceedings of Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM* (1998), vol. 2, IEEE Press, pp. 419–429.

[22] Woo, T. Y. C., and Lam, S. S. A framework for distributed authorization. In *Proceedings of the 1st ACM conference on Computer and communications security* (1993), ACM Press, pp. 112–118.

[23] Zurko, M., Simon, R., and Sanfilippo, T. A user-centered, modular authorization service built on an RBAC foundation. In *Proceedings of the IEEE Symposium on Security and Privacy* (1999), IEEE Press, pp. 57–71.

Appendix

A A Primer on Web Services and Business Processes

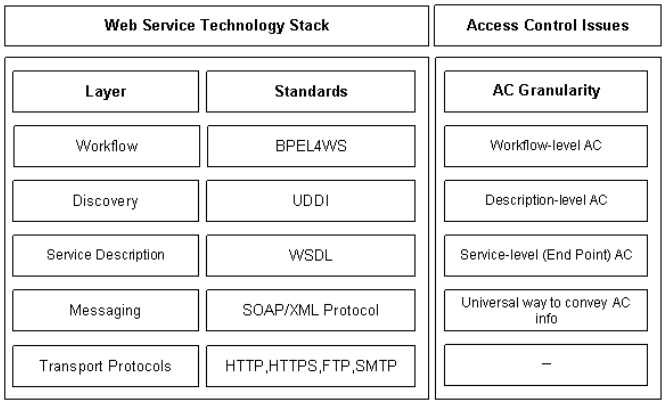


Figure 11: Web Services Technology Stack & Access Control Issues

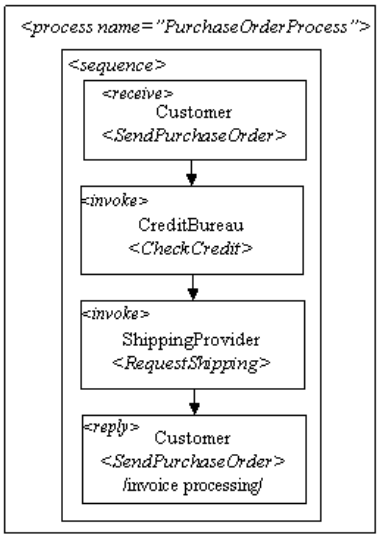
A Web Service as defined by IBM is: an interface that describes a collection of operations that are network-accessible through standardized XML messaging. A Web service is described using a standard, formal XML notion, called its *service description*. It covers all the details necessary to interact 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.

The WS architecture, as defined by W3C, is divided into five layers grouped into three main components - Wire, Description, and Discovery (Figure 11). 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). Moving forward we found the *Service Description* layer and the *Business Process Orchestration* layer. The service description layer 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 a business activity and primitive nodes are in WSDL. The recently released standard at this layer is the Business Process Execution Language for WS (BPEL4WS).

The BPEL4WS primitive activities are the following:

`<invoke>` invoking an operation on some Web service;
`<receive>` waiting for an operation to be invoked by someone externally;
`<reply>` generating the response of an input/output operation;




```
<process>
    <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 12: Example of BPEL4WS Process

`<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 12: 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 continue 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 12 is a very 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 executed in this order. The node contents is self explanatory.