Towards Proactive Policies supporting Event-based Task Delegation

Khaled Gaaloul, Philip Miseldine, François Charoy

Towards Proactive Policies supporting Event-based Task Delegation

Khaled Gaaloul, Philip Miseldine
SAP Research
Vincenz-Priessnitz-Strasse 1, 76131 Karlsruhe, Germany
Email: khaled.gaaloul@sap.com, philip.miseldine@sap.com

François Charoy
LORIA - INRIA - CNRS - UMR 7503
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
Email: charoy@loria.fr

***Abstract*—Delegation mechanisms are receiving increasing interest from the research community. Task delegation is a mechanism that supports organisational flexibility in the human-centric workflow systems, and ensures delegation of authority in access control systems. In this paper, we consider task delegation as an advanced security mechanism supporting policy decision. We define an approach to support dynamic delegation of authority within an access control framework. The novelty consists of reasoning on authorisation dependently on task delegation events, and specifies them in terms of delegation policies. When one of these events changes, our access policy decision may change proactively implying dynamic delegation of authority. Existing work on access control systems remain stateless and do not consider this perspective. We highlight such limitations, and propose a task delegation framework to support proactive enforcement of delegation policies.**

Keywords: Workflow, Delegation, Access Control, Policy.

I. Introduction

Many of the complex day to day applications in large organisations are conducted using workflow management systems. Workflow systems automate the management and coordination of organisational or business processes. A workflow typically comprises a set of coordinated activities, known as tasks. We currently observe a move away from predefined strict workflow modelling towards approaches supporting flexibility on the organisational level. One specific approach is that of task delegation [1]. We define task delegation to support organisational flexibility in the human-centric workflow systems, and to ensure delegation of authority in access control systems.

Typically, organisations establish a set of security policies, that regulate how the business process and resources should be managed [2]. While a simple policy may specify which user can be assigned to execute a task, a complex policy may specify additional authorisation constraints supporting delegation. Any mechanism that is used to support task delegation is based on workflow specifications and user authorisation information. Delegation authorisation constraints are defined as events on the control-, data- and task assignment layers of a workflow [1]. Hence, secure task delegation implies the presence of a fixed set of delegation events specifying delegation of authority and enforcing access control policies.

Most of the work done in the area of security constraints and access rights requirements does not treat delegation in sufficient details. On one hand, existing work in the domain of role based-access control are relatively coarse-grained and lack of flexibility [9]. On the other hand, existing work on authorisation decision making remain stateless and do not consider dynamic enforcement of policies [11] and [5]. At present, responses arising from access control requests are stateless such that a response is given to a particular request which is valid and true only at the time the request is made. If, however, this response changes due to a policy adaptation, no mechanism currently exists that allows the new response to be conveyed to the original requestor proactively. Such a mechanism is vital for supporting dynamic delegation of authority. When delegating a task, often the reasoning behind this is dependent on transient conditions (events). When one of these conditions changes during execution, our access policy decision may change: what was once acceptable for the delegatee to access, may not be suitable once the delegation is revoked. We do believe that delegation events define dynamic constraints for authorisation decision making that should not be neglected in advanced security mechanisms supporting task delegation.

The contribution of the paper is the definition of a dynamic delegation of authority approach to support proactive policies in access control systems. First, we motivate our work by introducing a real world scenario supporting delegation within a workflow. We then identify specific events-based task delegation model that would enforce policy changes. The novelty consists of separating the various aspects of delegation with regards to users, tasks and events, and specifies them in terms of delegation policies. Defined policies will regulate delegation control flow within a process while ensuring dynamic delegation of authority. Further, we present existing access control framework for authorisation decision making and discuss their functionalities and limitations. Finally, we present our approach that permits proactive enforcement of delegation policies, and discuss its integration into existing access control systems.

The remainder of this paper is organised as follows. Section 2 presents an e-government case study and motivates a delegation scenario. In section 3 we define a task delegation model, and present the security requirements for our approach. Section 4 presents our delegation framework for access control systems. Section 5 discusses related work. In section 6 we conclude and outline several topics of potential future work.

II. Context and Problem Statement

A. Motivating Example

To understand the motivation of our research, we present a real world scenario supporting task delegation. Mutual Legal Assistance (MLA) defines a workflow scenario involving national authorities of two European countries regarding the execution of measures for protection of a witness in a criminal proceeding. Here we describe the MLA process part in the Eurojust organisation A, where a Eurojust member (Prosecutor) receives the request of assistance from the Europol member in order to process it and send it to the concerned authority in country B (see Figure 1). User Alice member of role Prosecutor is assigned to execute the MLA process in Eurojust A. Activities that are part of the process are represented as tasks.

We applied the Business Process Modeling Notation (BPMN) to the MLA process. BPMN has emerged as a standard notation for capturing business processes, especially at the level of domain analysis and high-level systems design [6]. BPMN supports so called artifacts that enrich the process model by information entities that do not affect the underlying control flow and define extension points to add additional information to the model. We provided an extension for security semantics to leverage the specification of access control security policies, by adding authorisation constraints artifact regarding users authorised to perform a task.

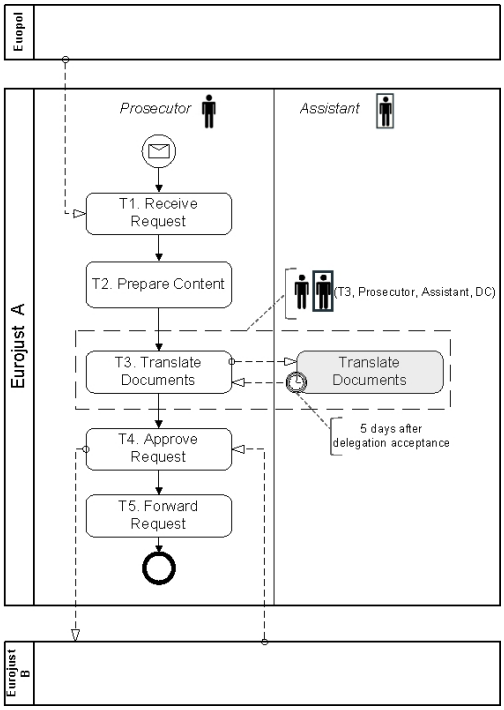

Fig. 1. **MLA delegation scenario**

In this scenario, the task "Translate Documents" T3 is originally only accessible by Prosecutor Alice, a fact defined in the workflow policy. This task is a long-running task and is expected to take 5 working days to complete. Alice is unavailable to execute this task due to illness, and will delegate it to User Bob. Bob is a member of role Assistant and is a subordinate to Prosecutor in the organisation hierarchy. Task delegation is a suitable approach to support organisational flexibility and to ensure alternative scenarios [1].

A policy can be defined as a level of defining access to task resources. We define an authorisation policy P for the MLA process. During delegation, the policy P is updated so that User Bob is now allowed to complete task T3. As such, Alice and Bob are here the delegator and the delegatee, respectively. Bob claims the task, and issues an access control request, is granted access, and executes the task. After two days, Alice interrupts her sick leave and returns to work. Once again, Alice is able to claim the task. Due to qualification considerations, it is decided that Alice should complete the task, and that Bob should revoke his actions, and free the task. The policy P need to be updated to reflect that only Alice has access to the task. As such, the original request made by Bob would now evaluate to a deny decision for access.

In traditional access control frameworks no mechanism exists that would alert User Bob to this fact automatically. Accordingly, it is not possible to revoke a previous response given to an access control request. Moreover, a manual review of the current access control rights and task executions is costly, labor intensive, and prone to errors. With a proactive mechanism, when the policy changes to reflect delegation events, the delegatee will be informed proactively. This inquires the need to support specific interactions on the access control architecture that they run on. Specific interactions are meant to be task delegation events that would be automatically captured, and conveyed back to the requestor for appropriate actions.

B. Discussion

We leverage a role-based delegation model to support human-centric interactions in the context of long-running tasks [3]. We assume that task execution is atomic and delegation authority is exclusively granted to the delegatee. Therefore, delegation criteria such as cascaded and/or partial are not considered in this paper [4]. On a delegation event, we define a task delegation relation as follows:

Definition 1. We define a task delegation relation $RD = (T,u_1,u_2,DC)$, where T is the delegated task, u_1 the delegator, u_2 the delegatee, and DC the delegation constraints.

Delegation constraints refer to the right of delegating accordingly to the workflow policy. For instance, DC is based on the role hierarchy (RH) definition in Eurojust, where the Assistant Bob is a subordinate to the Prosecutor Alice. In addition, DC refers to a temporary delegation, where a delegation period must be defined. Bob is not allowed to exceed T3 deadline (5 working days). We define the delegation relation for T3: $RD = (T3,Alice,Bob,(RH,5\ days))$.

Proactive policy enforcement mechanism is vital for supporting delegation in long-running tasks. This inquires the need to support specific interactions based on delegation events that would be automatically captured, and conveyed back to the requestor for appropriate actions within the access control system. At present, we can enforce delegation access rights via policy adaptation (i.e. permitting the delegatee to perform the delegated tasks). Subsequently, we update the *RD* specification in the workflow policy once a delegation event is triggered. It consists of adding a new policy authorisation constraint for the delegated user. If this constraint changes due to a policy adaptation (e.g., a task revocation event), a new response need to be conveyed to the delegator proactively.

III. Task Delegation Policies based-events

In this section, our concern is to identify events that would enforce policy changes. First, we present our task delegation model, and define transitions as events ruling task life cycle. We then specify from defined events delegation policies that would change during task execution. We argue that task delegation implies dynamic delegation of authority: when one of these events occurs, our access policy decision may change.

A. Task Delegation Model (TDM)

In recent work we defined a task delegation model (TDM) based on task life cycle specifications in the workflow management coalition [1], [15]. Figure 2 depicts a UML state diagram that illustrates the life cycle of our TDM in the form of a state transition diagram from the time that a task is created through to final completion, cancellation or failure. It can be seen that there are series of potential states that comprise this process. A task, once created, is generally assigned to a user. The assigned user can choose to start it immediately or to delegate it. Delegation depends on the assignment transition, where the assigned user has the authority to delegate the task to a delegatee in order to act on his behalf.

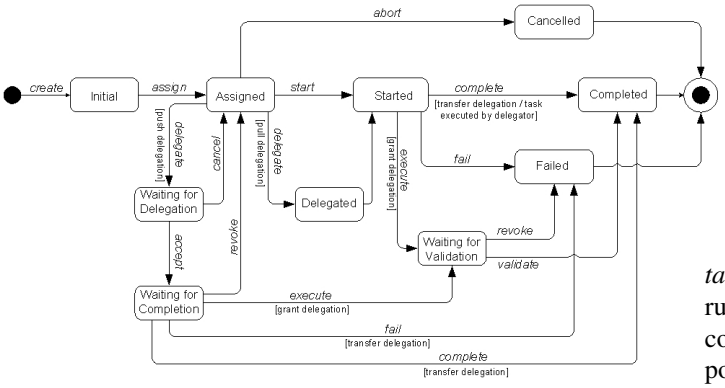

Fig. 2. **Task delegation model**

Intermediate events define controlled delegation within a workflow (e.g., *delegate, cancel, revoke*). For instance, the delegator might want to cancel. Our TDM would then go back to the previous state (*Assigned* state). The delegation control flow behaviour remains internal according to the task model, where *Completed*, *Cancelled* and *Failed* are the final states.

B. Security Requirement Statements

We define delegation transitions as events ruling delegation behaviour. We enriched TDM with additional events supporting delegation requirements such as pull/push mode and grant/transfer kind. The internal behaviour based-events may be a source to a policy change, thereby introducing advanced security requirements in access control enforcement systems. For instance, events like *accept* or *revoke* inquire access control enforcement supporting delegatee privileges. From the TDM, we analyse security requirements that need to be taken into account to define delegation policies based-events (see Figure 2).

- **Delegation mode**: It defines how delegation request is issued. Pull mode assumes that a delegator has at his disposal a pool of delegatees to be selected to work on his behalf. Push mode assumes that a delegator is waiting for an acceptance from a potential delegatee. Derived events from push mode are *accept*, *cancel* and *revoke*.
- **Delegation kind**: It may be classified into grant or transfer [16]. A grant delegation model allows a delegated access right (privileges) to be available for both delegator and delegatee. As such, the delegator is still having the control to *validate* or *revoke* the task, and the delegatee to *execute* it. However, in transfer delegation models, the ability to use a delegated access right is transferred to the delegatee; in particular, the delegated access right is no longer available to the delegator. There is no validation required and the task is terminated (*complete*/*fail*) by the delegatee.
- **Delegation of authority**: It permits to a delegator to assign a subset of his assigned privileges to a delegatee who currently do not possess the required authorisation to execute the task. For instance, *delegate* is an event that will trigger task delegation, thereby updating a policy to enforce access control for a delegatee.
- **Access control enforcement**: It permits dynamic policy enforcement. For instance, *revoke* implies the revocation of delegated privileges where the delegator will take the control back on his assigned task and, therefore, cancel the previous policy decision.

Definition 2. We define a policy P = *(target,rule,C)*, where *target* defines where a policy is applicable, *rule* is a set of rules that defines the policy decision result, and C the policy constraints set that validates the policy rule. A delegation policy is a policy P_D = $(target_D, rule_D, C_D)$, where $target_D$ = *RD*, $rule_D \subseteq rule$ and $C_D \subset C$ and $C_D = CD \bigcup events$.

The specified *events* define the condition to validate the policy decision effect. An event change may inquire a policy decision change. In the following, we classify delegation events and identify the relationship between delegation events, delegation criteria and policies decision change (see Table I).

TABLE I
DELEGATION POLICIES CHANGES BASED-TASK EVENTS

Delegation Events	Push Delegation		Pull Delegation		Policy Decision Change
	Grant	Transfer	Grant	Transfer	
delegate	✓	✓	✓	✓	✓
accept	✓	✓	*x*	*x*	✓
cancel	✓	✓	*x*	*x*	*x*
execute	✓	*x*	✓	*x*	*x*
validate	✓	*x*	✓	*x*	✓
revoke	✓	*x*	✓	*x*	✓
fail	*x*	✓	*x*	✓	*x*
complete	*x*	✓	*x*	✓	*x*

We do believe that events such as *accept* or *validate* are a part of delegation policies and have a direct impact on delegated authority, thereby inquiring dynamic enforcement of policies.

We present two examples to explain Table I:

Example 1: *validate* event is defined in both push and pull modes. It supports grant delegation, where a delegatee need to wait for the validation from the delegator. This event will enforce a policy change and terminate authorisation policy for the delegatee. The validation leads to the completion of the task, and the revocation of the delegated privileges. For instance, Bob work is validated by Alice and then his delegated authority is no more valid in the policy.

Example 2: *fail* event is defined in both push and pull modes. It supports transfer delegation, where a delegatee terminates the task by himself without validation. Defined policy will take effect until the termination of the task during transfer delegation, where no new updates are required since all the task privileges are transferred to the delegatee.

Returning to the example, we can observe a dynamic policy enforcement during delegation. Initially, T3 is delegated to Bob and the delegation policy for T3: P_D = *(RD,permit,(RH, 5 days,delegate))*. In the meanwhile, User Alice is back to work before delegation is done and would cancel what was performed by User Bob so far. Alice is once again able to claim the task and will cancel the policy effect (permit) for Bob. The event *revoke* will be updated in the policy, and a notification (deny) is then conveyed back to Bob for appropriate actions. Thus, the delegation policy for T3 need to be updated and P_D = *(RD,deny,revoke)*.

IV. A SECURE FRAMEWORK FOR TASK DELEGATION

In this section, we develop a framework to support secure task delegation. We present a modular architecture ensuring dynamic delegation of authority and show how proactive policy decisions will be implemented on existing access control frameworks (ACF). In the context of delegation, when a request is issued, it is stored along with details of how to inform the requestor (the delegatee) if the policy decision to the request changes. When a policy is changed, previous requests are re-evaluated, and the requestor is informed that his access rights have changed. To support this approach, we propose an extension to an abstract ACF architecture that permits proactive enforcement of policies, based on delegation events that would alter previous policy decisions (see policies changes identified in Table I).

A. Architecture Overview

We describe the main components of the task delegation framework supporting proactive policy enforcement. We detail what parts were changed in which way and what the new extended architecture looks like (see Figure 3).

- **Policy Manager**: It allows an administrator to define policies. Through a graphical user interface, the administrator can navigate through the policy document, select document elements (e.g., targets, authorisation rules, obligations), and specify values for selected elements. For instance, an administrator defines an authorisation policy *P* with decision *permit* on target task T3 for subject Alice with role Prosecutor. In the context of delegation, we assume that a delegator is allowed to administrate policies, thereby defining delegation rules. Delegation policies will embed delegatees attributes for authentication and authorisation purposes.
- **ACF**: An access control framework (ACF) is defined as a set of software components that accept requests to access resources, analyse these against policies representing actual access rights to resources, and return a response based on this analysis. To illustrate the original architecture of an ACF without extension, a request is issued by the requestor, which is received by the *Receiver* component in ACF. This is then sent to the *Analyser* component that queries policies stored in a policy database. A response is generated by the *Responder* component, which defines a decision (permit, deny, or notapplicable) that is sent back to the requestor. It should be noted, that the above appears asynchronous for the requestor; they provide the request, and a response is produced.
- **Dynamic Policy Enforcement**: It defines our approach to support proactive policy decisions. We extend the ACF architecture with additional components related to policy database. When the *Receiver* receives a request, it sends this to a Request Database that stores this request. A *Policy Adaptation Listener* component polls the *Policy*

Database and sends an event to a *Re-evaluator* component when a policy has changed (see delegation events that change policy in Table I). This queries the *Request Database* to retrieve the previous requests made, and sends this to the *Analyser* component for re-evaluation. The *Analyser* then sends back a new response to the *Re-evaluator*, which queries the *Request Database* to see if this is different to the response given to the request being analysed. If this is a different response, the *Invocation Manager* component invokes the "contact point" provided by the requestor (and stored in the Request object) with the new response.

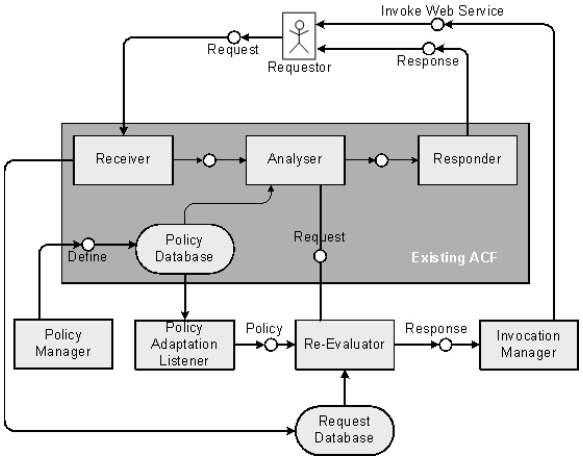

Fig. 3. **Architectural extensions supporting delegation policies**

B. Architecture Requirements

On an architectural level, as requests are required to be re-evaluated upon a policy change, a storage mechanism of previous requests and the given response is needed. If a previous request is re-evaluated and a different response to the one stored is produced, the ACF must inform the requestor of the new result. Thus, a mechanism must exist that triggers a re-evaluation when it detects a policy change. These effects of the policy change would be automatically captured, and conveyed back to the requestor for appropriate actions. In addition, an invocation component is needed that actually marshals this information to the requestor.

On a language level, the approach would require new constructs to describe the invocation method that the ACF can use to contact the requestor. As such, this acts as a "contact point" for the requestor. In a service-oriented architecture (SOA), this contact point could consist of the endpoint of a service that could be invoked (see the Invoke Web Service in Figure 3). Subsequently, all access policies must be centralised and referenced by the SOA architecture which is protected. We give an SOA a single point of access and we let the services register with our ACF. Since services are essentially black boxes, we define how to contact them and to sort out what it means when policy changes (i.e., proactive policies).

On a technical level, the *Policy Manager* generates policies and subsequently embed credentials attributes for authentication and authorisation purposes [1]. Credentials providers such as certification of authorities issue digital certificates to the requestor in order to compute his request by the ACF. At this stage, the *Receiver* component acts as a policy enforcement point to perform access control by making decision request and enforcing decisions. For instance, an X.509 Attribute Certificate (AC) is issued to the delegatee for authentication and authorisation purposes. AC will ensure integrity, protection and non-repudiation through a digital signature. The *Receiver* gets his attributes certificate and checks his permissions afterwards. The retrieved attributes are validated against the policy (e.g., subject attributes, validity time). Once the delegatee has been successfully authenticated he will attempt to perform specified actions on task resources. At each attempt, the *Receiver* passes the access request to the *Analyser* to decide. Decisions results (permit, deny, or notapplicable) are then sent via the *Responder*.

A new re-evaluation of a policy defines new AC for further request with regards to policy changes. For instance, a revocation implies the cancellation of the issued AC for the delegatee previously. Currently, techniques like temporary certificates or certificate revocation list (CRL) are time-based, and, therefore, do not fulfill event-based requirements. As a first solution, a service is invoked to contact the delegatee and based on the mutual agreement between delegation principals, appropriate actions will take place (e.g., the cancellation of his work and the log off from the system).

V. Related Work

The eXtensible Access Control Markup Language (XACML) was developed in order to provide a uniform way of specifying access control policies in XML [13]. Policies comprising Rules, possibly restricted by Conditions, may be specified and targeted at Resources, Subjects and Actions. Resources, Subjects, Actions and Conditions are matched with information in an authorisation request context using attribute values and a rich set of value-matching functions. The outcome or Effect of a policy evaluation may be Permit, Deny, NotApplicable or Indeterminate. The current XACML standard does not provide explicit support for task delegation.

Chadwick et al. [11] proposed a solution based on the XACML conceptual and data flow models supporting dynamic delegation of authority. The approach describes a new conceptual entity working alongside the XACML policy decision point in the authorisation decision making. The proposed architecture offers a flexible and dynamic way to manage users credentials and assert them in the remote credential providers, however this is not enough to support dynamic delegation of authority. We do believe that delegating a task requires more effort and involves additional specifications related to task delegation events. Our TDM provides the means to determine faithfully delegation polices in-line, thereby ensuring proactive policy decisions when events such as revoke and validate are triggered during task execution.

Seitz et al. [9] investigated how an authorisation management system based on XACML can be extended to use flexible delegation mechanisms. They developed a separate policy administration point component called Delegent that specifies allowed modifications on different elements of an XACML policy for different users. This work was later extended in a product called Axiomatics to support delegent authorisation system [14]. Authors investigate the administration side of the policy while overriding access control in XACML. Administrating delegation policies is, however, stateless and lacks of reactivity to support policy change when delegating a task. We need a reactive approach to reflect events change and support dynamic enforcement of policies.

PERMIS [8] is a middleware authorisation framework, which focuses mainly on role based access control (RBAC) model. PERMIS XML policy can be extended to support additional conditions such as role assignment validity, delegation depth and target access clauses. The PERMIS framework does not provide direct support for bilateral exchange of policies and credentials to address privacy issue and trust. This is not enough to manage security for systems in which organisations are dynamically built with the collaboration of multiple independent organisations sharing their resources. This is especially the case when we consider authorisation decision making supporting delegation policies involving ad-hoc as well as process-based human interactions cross-organisations.

VI. Conclusion and Future Work

Providing access control mechanisms to support dynamic delegation of authority via proactive policy enforcement, is a non-trivial task to model and engineer. In this paper we have presented problems and requirements that such a model demands, and have architected a solution based on existing access control frameworks (ACFs). We have also presented at a conceptual level the different components that are necessary to support dynamic policy enforcement. The motivation of this direction is based on real world processes from an e-government case study, where a long-running task may support changes during delegation. Delegation policies may change according to specific events. We define the nature of events based on task, and describe their interactions with policies decisions. When relevant events occur, we define how they are detected and how to interact with them. In this context, we proposed an extension to an abstracted ACF architecture that permits delegation policies enforcement, thereby ensuring dynamic delegation of authority.

The next stage of our work is the implementation of our framework using XACML standard. We propose an extension to XACML specifications supporting task delegation requirements. We then integrate the proactive policy enforcer in the existing architecture. Future work will look also at enriching our approach with additional delegation constraints supporting historical records. Delegation history will be used to record delegation that have been made to address administrative requirements such as auditing.

References

[1] Khaled Gaaloul, Andreas Schaad, Ulrich Flegel, and François Charoy. A Secure Task Delegation Model for Workflows. In *SECURWARE 2008: The Second International Conference on Emerging Security Information Systems and Technologies*, Cap Esterel, France, August 25-31, 2008. IEEE Computer Society.

[2] Vijayalakshmi Atluri and Janice Warner. Supporting conditional delegation in secure workflow management systems. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies, Stockholm, Sweden*, ACM press 2005.

[3] E. Barka and R. Sandhu. Framework for role-based delegation models. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, page 168, Washington, DC, USA, 2000. IEEE Computer Society.

[4] L. Zhang, G.-J. Ahn, and B.-T. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.

[5] Elisa Bertino, Silvana Castano, Elena Ferrari, and Marco Mesiti. Specifying and enforcing access control policies for xml document sources. pages 139–151, Hingham, MA, USA, 2000. Kluwer Academic Publishers.

[6] The Workflow Management Coalition. Process Definition Interface XML Process Definition Language (2005). http://www.wfmc.org.

[7] Babak Sadighi Firozabadi, Marek J. Sergot, and Olav L. Bandmann. Using authority certificates to create management structures. In *Security Protocols, 9th International Workshop, Cambridge, UK, April 25-27, 2001, Revised Papers*, pages 134–145.

[8] David W. Chadwick and Alexander Otenko. The permis x.509 role based privilege management infrastructure. *Future Generation Comp. Syst.*, 19(2):277–289, 2003.

[9] Ludwig Seitz, Erik Rissanen, Thomas Sandholm, Babak Sadighi Firozabadi, and Olle Mulmo. Policy administration control and delegation using xacml and delegent. In *6th IEEE/ACM International Conference on Grid Computing (GRID 2005), November 13-14, 2005, Seattle, Washington, USA, Proceedings*, pages 49–54, 2005.

[10] Tuan-Anh Nguyen, Linying Su, George Inman, and David W. Chadwick. Flexible and manageable delegation of authority in rbac. In *21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, Volume 2, May 21-23, 2007, Niagara Falls, Canada*, pages 453–458, 2007.

[11] David W. Chadwick, Sassa Otenko, and Tuan-Anh Nguyen. Adding support to xacml for dynamic delegation of authority in multiple domains. In *Communications and Multimedia Security, 10th IFIP TC-6 TC-11 International Conference, CMS 2006, Heraklion, Crete, Greece, October 19-21, 2006, Proceedings*, pages 67–86, 2006.

[12] Andreas Schaad. A Framework for Organisational Control Principles. PhD thesis, The University of York, York, England, 2003.

[13] eXtensible Access Control Markup Language (XACML v2.0). Standard, Organization for the Advancement of Structured Information Standards (OASIS), February 2005. http://docs.oasis-open.org/xacml/2.0/access-control-xacml-2.0-core-spec-os.pdf.

[14] Axiomatics, Delegent authorisation system. OASIS XACML InterOp Demo. RSA Conference 2008 San Francisco, California, USA. 7-11 April 2008.

[15] The Workflow Management Coalition. Workflow Management Coalition Terminology and Glossary. Document Number WFMC-TC-1011, February 1999.

[16] Jason Crampton and Hemanth Khambhammettu. Delegation in role-based access control. In *Proceedings of the Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006*, Lecture Notes in Computer Science, pages 174–191. Springer, 2006.