Delegation Enabled Provenance-based Access Control Model

Noha Nagy Mohy
Information System Dept,
Faculty of Computers and Information, Cairo University,
Giza, Egypt
n.nagy@fci-cu.edu.eg

Hoda M. O. Mokhtar
Information System Dept,
Faculty of Computers and Information, Cairo University,
Giza, Egypt
h.mokhtar@fci-cu.edu.eg

Mohamed E. El-Sharkawi
Information System Dept,
Faculty of Computers and Information,Cairo University,
Giza, Egypt
m.elsharkawi@fci-cu.edu.eg

Abstract—**Any organization aims to achieve its business objectives, secure its information, and conforms to policies and regulations. Provenance can help organizations achieve these goals. As provenance stores the history of the organization's workflow, it can be used for auditing, compliance, checking errors and securing the business. Provenance Based Access Control (PBAC) is one of the new access control models that used to secure data based on its provenance. This paper introduces Delegation Provenance based Access Control (DPBAC) model that accounts for the delegation of access rights and also introduce an extension to the Open Provenance Model (OPM) in order to store the history of the delegation to be used for auditing purposes.**

Keywords— Provenance, access control, delegation, OPM.

I. INTRODUCTION

Provenance (also referred to as lineage or pedigree) is typically defined as "the history of data and derivation" [1]. Provenance is usually represented using the Open Provenance Model (OPM)[1] that is considered a standard for representing provenance. Provenance has gain much research attention research in recent years as it became a popular and an essential requirement in many applications such as databases and data warehouses [2], scientific workflows [3], [4], grid computing [5], cloud computing [7-9] , social networks [6]–[8], geographic information systems [9] and conformance and auditing process [10]–[18].
Provenance plays a major role in achieving one of the main goals of any organization namely, data security.

Traditional access control models such as Discretionary Access Control (DAC) [2], Mandatory Access Control (MAC) [2] and Role-Based Access Control (RBAC) [3] models no longer satisfy the main requirement for modern enterprise environments which is dynamic policies. So, other models are needed to meet these requirements. Provenance Based Access Control (PBAC) [19] is an access control model that is used to secure traditional data by specifying access control policies based on its provenance.
Delegation of access rights is one of the main facilities that is provided in access control models. It has been studied extensively in research [20]–[25] and most of them depend on RBAC. Although, RBAC provides an easy management, it suffers from multiple issues such as: (i) *scalability problem* as the policy become finer grained it needs to separate role for each combination of attribute values; (ii) *the policy used is static* where the resources and the users must be identified beforehand; (iii) *RBAC doesn't permit constraints on resources, history of processes or environment attributes*; (iv) *most modern application are dynamic systems* as we don't have a predefined roles or permission and the access control specification is dynamic in terms of history of users or tasks. All these issues motivate researchers to finds alternative access control models that attempt to solve these issues. Provenance Based Access Control (PBAC) is considered one of the access control models that was proposed to meet the requirements of modern systems, however, delegation function was not introduced in this model. Inspired by the importance of delegation in many real world applications, in this work we try to extend this model to incorporate delegation function in a secure manner.
In summary, the general expectations of providing delegation in PBAC are: (i) support fine grained constraints over history of delegator, delegatee or task being delegated; (ii) specifying delegation constraints using regular expressions where same policy rule applied to multiple instances; (iii) specifying delegation condition considering workflow constraints such as ordering of operation, Separation of Duty (SOD)[26] and Binding of Duty (BOD)[26]; (iv) represent the history of delegation in OPM graph to be used for specifying constraints and auditing purposes.

The main contributions of this paper are:

- Providing an extended model for PBAC to be delegation enabled.
- Providing a mechanism to ensure secure delegation of the access rights while preventing colluding users from acquiring more rights through delegation by using provenance.
- Introducing new notations to the OPM to store delegation history to be used further for specifying constraints or for auditing purposes.

The remainder of this paper is organized as follows: Section 2 describes the related work. Section 3 presents the required background about the main concepts used in the paper which are provenance, PBAC Model and delegation. The proposed model and its components are described in section 4. Finally, section5 concludes the paper.

II. Related Work

Recently, many research efforts have considered securing provenance. Researchers try to use the regular access control models to control access to data and provenance. Nevertheless, these models suffer from ignoring the relationship between data and its provenance. They simply use access control for data and access control for the provenance separately. The authors in [27] specify two types of provenance security control models which are Provenance Access Control (PAC) and Provenance-Based Access Control (PBAC).

Role-Based Access Control (RBAC) model groups users with similar permissions or responsibilities into roles [28]. Various delegation models based on RBAC were proposed. The authors in [20] identify the important issues to be considered for delegation in RBAC and proposed Role-Based Delegation Model Zero (RBDM0). The authors in [23] proposed Role-Based Delegation Model One (RBDM1) which is an extension of RBDM0. The model adds hierarchical relationships between the roles. It uses this hierarchy to describe which roles can delegate permissions and which ones can acquire delegation permissions. The authors in [29] introduce a delegation model that considers the privacy policies in delegation decision process. The work presented in [24] studied grant, transfer delegation for RBAC and the enforcement and revocation of delegation process in the context of workflow systems. The authors in [30] proposed a delegation model Attribute-Based Delegation Model (ABDM). They extend the delegation constraints to be delegation prerequisite condition and delegation attributes expressions. As we presented most of the previous models depend mainly on roles. The work in [31] is very related to our proposed work as the authors discuss the problem of providing delegation in a secure manner but they also depend on static defined roles. They proposed a new mechanism that uses source based enforcement mechanism.

The new mechanism requires that each user must specify the privileges to be used and the source of the privileges. When there is a conflict of interests between two different steps in the workflow they check the conflicts between the sources of the delegation not between the delegated users. There were no discussions about other techniques that do not depend on the roles.

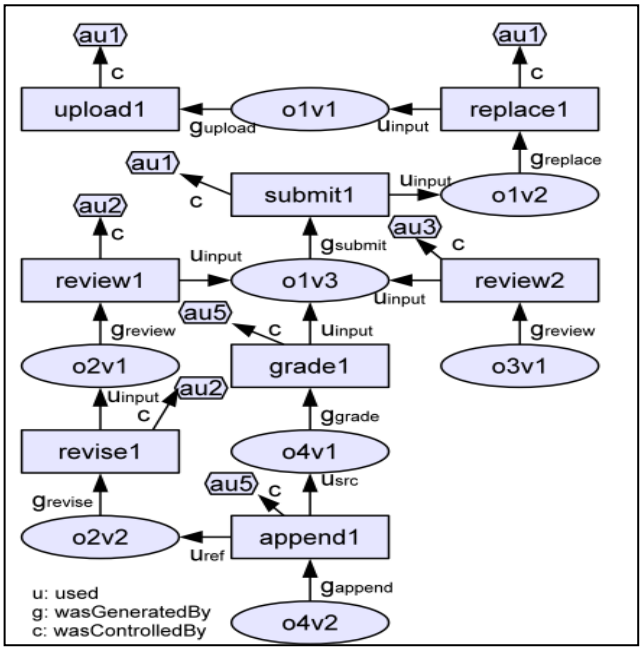

Fig 1.Homework Grading System in OPM.

III. Background

This section presents an overview about the main concepts and terms about provenance and delegation which are relevant for the remainder of this paper.

A. Provenance

Provenance is used to know more about the system. So, we can use it as a way to control the system's behavior [32]. Securing provenance data has been studied in recent years but using provenance data to control access and secure traditional data is rarely discussed in literature [19].

Open Provenance Model (OPM) is considered the standard way to model the provenance data. It was proposed in 2008 [1]. It models provenance as a Directed Acyclic Graph (DAG). The nodes of the graph could be artifacts, processes or agents. Artifacts represent the data objects that are used by a task or generated from a task. Processes are the tasks or actions of the system. Agents represent the persons who perform the tasks. The arcs represent the relationships between the OPM nodes. In this paper we use the same example homework grading system introduced in [19] . Figure 1 presents a simple example for the homework grading system in OPM graph.

B. Delegation

Delegation is not a new concept it has already been studied specially in Role-Based Access Control (RBAC) models [33],26] where the delegation is based on roles and role hierarchy. The user may delegate a role with all its permissions or he can just delegate a specific permission in a role. Also, it has been studied in Task-Based Authorization Control (TBAC) [25] where Separation of Duty (SOD) and Binding of Duty (BOD) [34] are the most constraints that control the delegation process. Delegation process consists of the following five elements:

1. Delegator: is the user who performs a delegation
2. Permission: the right that is being delegated from one user to another.
3. Delegatee: the user who receives a delegation.
4. Delegation form: specify if the delegation is granted delegation or revoke the delegation.
5. Delegation conditions: it describes the conditions that govern the delegation operation.

In delegation the security policy should specify the level of the delegation whether it is single or multi-step delegation. Multi-step delegation permit the delegatee to grant the received permissions to others [35]. For allowing multi-step delegation the process will be more complex as the system needs to be sure that no cycles exists in the delegation chain. The delegation chain contains all the users who received the same grant permission.

IV. DELEGATION ENABLED PROVENANCE-BASED ACCESS CONTROL (DPBAC)

In this section, we introduce how to specify delegation constraints using regular expressions specified based on provenance data. Also, we discuss how to include delegation in OPM.

A. DPBAC Model

DPBAC extends the capabilities of PBAC proposed in [19] with the power of delegation function. It is used not only to specify access rights, but also to specify delegation authorization rules, and ensure secure delegation. In our proposed model we make the following assumptions: First, we only consider user delegation which is temporary and dynamic delegation that is limited by time and constructed at runtime. Second, we ignore the scenario where the delegatee refuses to acquire the delegated permission. We suppose that if the delegation is successful the delegatee is granted the delegated permission. Third, only the delegator is allowed to revoke the permission he delegated which make the revocation process more controllable.

a) DPBAC Model Components

DPBAC model is presented in Figure 2. We extend the architecture of PBAC by adding new components such as workflow rules, delegation policies (DP), and extended OPM that stores delegation history. This section presents the DPBA components.

Acting Users (AU): are the users who request to execute a specific task on an object.

Actions (A): are the tasks requested by the acting user to do with a specific object.

Objects (O): data objects that users request to perform specific actions on it.

Request: There are three types of the user's requests. Grant request, revoke request, and request for access permission.

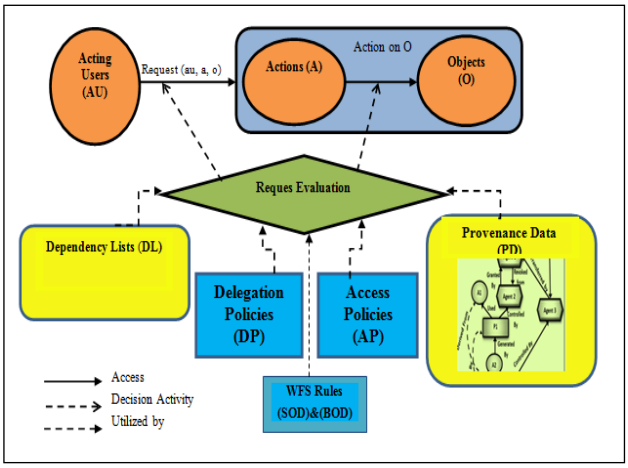

Fig 2. The Components of DPBAC.

Dependency Lists (DL): it consists of dependency path expressions of the provenance graph represented in OPM. It is used to make the process of policy specification more flexible. It is defined based on the navigation of the OPM graph relationships.

Policies: are rules that evaluate user's requests. These policies can be specified based on the defined dependency lists. There are two types of policy the access policy (AP) and the delegation policy (DP) that will be introduced later.

Workflow Rules: contain all the regulations that regulate the implementation of the workflow tasks with respect to SOD and BOD.

In our discussion "separation of duty" is a constraint that requires that conflicted tasks must be executed by different actors.

While, "binding of duty" is a constraint that states that some tasks must be executed by the same actors.

Provenance Data (PD): Captures the workflow executed tasks and represents it in OPM graph. The next subsection

will show how to capture grant and revoke transactions in the provenance OPM graph.

B. Provenance Data in DPBAC Model

Provenance data is represented in the enhanced OPM described in [36]. The proposed model uses this enhanced OPM and extends it with two new special types of OPM processes notations that are used to track the delegation history. The new processes are "Grant", and "Revoke", those processes connect two agents nodes. Also, we introduce new types of relations "GrantedBy"," RevokedFrom","GrantControl". These edges connect the Grant and Revoke processes with the agents. Figure 3 represents the extended OPM with delegation notations.

Process "Grant" stores attribute such as (Delegator, Delegatee, Delegated task, Object, delTime, Period, dLength). The "Delegator" is the agent who wants to grant one of his permission to another user. "Delegatee" is the agent who will acquire the permission from the delegator. "Delegated task" is the permission that delegated from the delegator to the delegatee. The "object" is the data object that will be used by the delegated task. "delTime" is the time of delegation. "Period" is the time period where the delegation will be valid during this period. "dLength" represents the delegation path length. We store the delegation length only when the application system permits multi-step delegation.
Process "Revoke" is the opposite of Grant. It stores the following attributes (Delegator, Delegatee, revTask, Object, revTime) where the "Delegator" is the agent who wants to revoke the granted permission. "Delegatee" is the agent who will lose the granted permission. "revTask" is the permission that is revoked from the delegatee. "Object" is the data object that accessed by the revTask. "revTime" is the time of revoking the delegated permission.

In Figure 3 the relation "GrantedBy" shows that Agent2 is a delegatee and he is the actor of the delegated permission P1. The relation "GrantControl" is used to define that Agent1 is the delegator. The relation "RevokedFrom" is used to state that Agent2 is no longer granted permission P1.

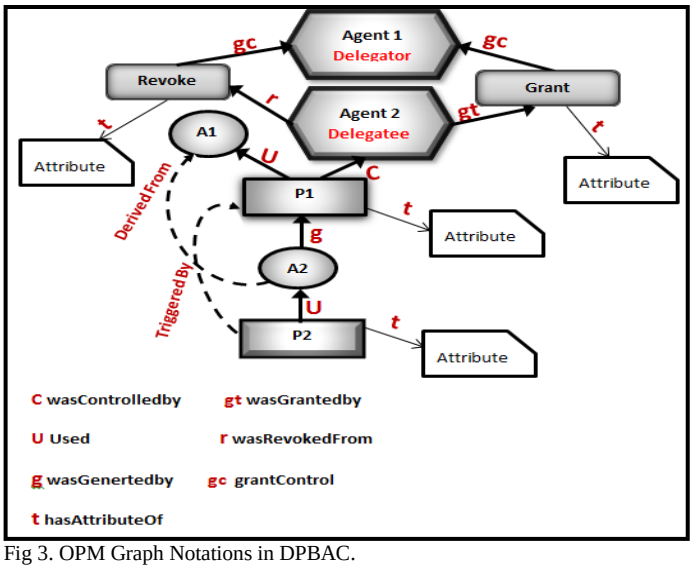

Fig 3. OPM Graph Notations in DPBAC.

C. Policies in DPBAC Model

DPBAC use provenance data to specify not only the access rules, but also delegation authorization rules. We categorize some scenarios where provenance can be used to specify conditional delegation.

Workload Delegation: Some applications specify maximum number of assignments for each user. This requires checking all the previous executed tasks and previous delegated tasks which are stored in provenance data to evaluate any upcoming requests from users.

History-based Delegation: The delegation constraints depend on the history of the delegator, the history of the delegatee, or the history of the object being delegated all the histories are stored in provenance data.

Delegation Considering Workflow Constraints: such as SOD, BOD, and the sequence of tasks. We can use provenance to check the history of the executed tasks on an object.

Multi-Step Delegation: Provenance can be used to ensure that no delegation cycles exist by checking the history of delegation permissions between users for a specific delegated task.

We extend the policies of PBAC to acquire the delegation policies (DP) that includes a set of rules used to evaluate granting access from one user to another, and for revoking access permission. In our homework grading example, we will present two examples of delegation policies.

Example 1: The author of the homework can delegate replacing his homework to another user only if the homework is not reviewed yet. This can be represented using the following policy.

$$Allow(User1,grant,User2,replace,Hw) \rightarrow User1 \in (Hw,wasAuthoredBy) \wedge |(Hw,wasReviewedof^{-1})| = 0.$$

Example 2: The reviewer can delegate his task of reviewing a homework to another one only if he didn't review it, the delegatee is not the author of the homework, the delegatee is not one the reviewers of this homework, the homework didn't reviewed more than 2 times, and the homework is not graded yet. This can be represented using the following policy:

$$Allow(User1,grant,User2,review,Hw) \rightarrow User1 \notin (Hw,wasAuthoredBy) \wedge User1 \notin (Hw,wasReviewedBy) \wedge User2 \notin (Hw,wasAuthoredBy) \wedge User2 \notin (Hw,wasReviewedBy) \wedge |(Hw,wasSubmittedof\text{-}1)| \neq 0 \wedge |(Hw,wasReviewedof\text{-}1)| < 3 \wedge |(Hw,wasGradedof\text{-}1)| = 0.$$

D. Request Evaluation in DPBAC Model

The request evaluation module that presented in figure2 is the main components of DPBAC. It worked at the execution time when a request is received. It is responsible for finding the appropriate policies (workflow rules, delegation policies, access policies), dependency list and provenance for the received request, Then it evaluates the request to ensure that all the policy conditions are satisfied.Moreover, DPBAC used to ensure secure delegation of permissions.

Satisfying policy conditions doesn't prevent colluding users from acquiring permission they are not authorized. As an example, check this scenario. Alice has the right to submit her loan request she delegated this permission to Michal. Michal submits the loan request. After that Alice requests to review the loan submitted by Michal. The workflow constraints state that submitting and reviewing a loan must be executed by different users. In our case this constraint is satisfied and Alice will review her loan.
DPBAC prevents this type of collusion as the evaluation of the delegation depends on the Historical Tasks function.

ALGORITHM 1: EVALUATE GRANT REQUEST

```
INPUTS: Delegator,Delegatee,Task,Object.
OUTPUTS: Permit or Reject Request
BEGIN
        Let DelPolicy=the delegation policy
        rules for this delegated task.

        Let CT = the conflicts TASKS from
        workflow rules

        Get historical tasks(ET,RT,DT) of the
        Delegatee

        IF no conflicts among ET,RT,GT and
        DelPolicy THEN

           IF provenance record not exists
           THEN    //single Delegation
                Permit request
           ELSE
             Let Dellist=all the previous
             delegatee and the actual delegator
               IF the delegatee is not one
               of the Dellist & length(Dellist)
               <=dlength THEN
                        Permit request
                ELSE    //cycle exists
                        Deny request
                ENDIF
             ENDIF
         ELSE // conflicts exists
                Deny request
         ENDIF
END
```

Definition3 (Historical Tasks): tasks that were executed by a particular user. In addition to all tasks that were granted from others to that particular user. Moreover, it contains all tasks that were granted by that user to others. And hence, we can formally divide the historical tasks into the following three groups:

- Executed Tasks (ET): all tasks performed by a specific user.
- Received Tasks (RT): all tasks acquired by delegation.
- Delegated Tasks (DT): all tasks that are granted to other users.

Back to the aforementioned scenario when Alice requested to review the loan submitted by Michal, the DPBAC will retrieve Alice historical tasks and search for any conflicts. The list of DT tasks for Alice contains a submit loan permission to Michal, which is conflicting with the requested task review. And hence, the request will be rejected. This means that any grant or access request that permit a user to perform conflicting tasks will be rejected according to the user's historical tasks.

As we mention before there are three types of the user's requests. We will discuss these types of requests in the following subsections.

a. Grant Request

The evaluation of the grant request is based on algorithm1. It requires three steps.

Step1 Delegator constraints: check if the delegator can delegate the permission. In other words, this means he has the right to access this permission and no policy rule prevents him from granting this permission.

Step2 Delegatee constraints: check if the delegatee can acquire the permission. This requires ensuring that the delegation process is secure and the new grant permission will not cause the delegatee to acquire more authority than he authorized to have using historical tasks of delegatee and workflow rules.

Step3 Other constraints: ensures that the object can be granted and all the policy rules of the granting permission are satisfied such as contextual constraints (time, location), and length of the delegation chain in case of multi-step delegation.

b. Revoke Request

The evaluation of this type of request requires two steps.

Step1: Check that the revoke is requested by the delegator of the permission being revoked.

Step2: check if the request requires single revoke or cascading revocation of delegated permission. Cascading revocation needs provenance data to track the history of delegation (delegation chain) and then revoke the permission from all the users within the delegation chain.

c. Access Request

The evaluation of access request requires taking into consideration the delegation process. DPBAC evaluate access requests based on algorithm 2 which require two steps.

Step1: check the user can access this request. This requires getting the historical tasks of the user and get all the policy rules related to the request if no conflicts and all the access policy constraints are satisfied then the request is permitted.

Step2: if this user is not permitted request, we need to check his delegated tasks (DT) as he may be delegated to execute the requested task. This also requires ensuring that the delegation is still valid. Otherwise the request is denied.

ALGORITHM 2: EVALUATE ACCESS REQUEST

```
INPUTS: User, Permission(Task, Object)

OUTPUTS: Permit or Deny

BEGIN
       Get user's historical tasks
       (ET,RT,DT)

       Get the policies and the workflow
       rules for the requested permission

       IF User ∈ Can_access(Task,O) ∧ no
       conflicts among historical taks
       (ET,RT,and DT) and Task THEN
              Permit request

       ELSE IF User∈grant (Task,O)∧
       (period   isvalid∧ (User ∉
       revoke(Task,O)THEN
            Permit Access
      ELSE
             Deny Access
       ENDIF
END
```

CONCLUSIONS AND FUTURE WORK

This paper explores some extensions to Provenance Based Access Control (PBAC) including multiple issues of delegation such as single delegation, multi-step delegation, and revocation process by keeping in mind workflow constraints. We extend the Open Provenance Model (OPM) by adding the concept of delegating access rights from one user to another. A new special type of processes (Grant, Revoke) and relationships ("Granted By", "RevokedFrom "," GrantControl") are introduced. The proposed model Delegation Enabled Provenance Based Access Control (DPBAC) uses dependency path pattern to define the delegation authorization rules. Also, we have introduced how to provide secure delegation by using provenance data to prevent collusion among users to do unauthorized actions using delegation by detecting the conflicts among the user rights, history of executed tasks, and the new rights delegated to him. We present how to use provenance to add more facilities to PBAC such as revoking permission after performing the delegated task, prevent delegation cycles when multi-step delegation is allowed. For future work some issues need to be investigated such as handling big amount of provenance data efficiency and using provenance data for auditing and checking compliance.

REFERENCES

[1] L. Moreau, J. Freire, and J. Futrelle, "The open provenance model: An overview," Proven. Annot. Data Process. Third Int. Proven. Annot. Work. IPAW, pp. 323–326, 2008.

[2] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi, "Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases," Futur. Gener. Comput. Syst., vol. 27, no. 6, pp. 781–789, 2011.

[3] D. Koop, E. Santos, B. Bauer, and M. Troyer, "Bridging workflow and data provenance using strong links," SSDBM, pp. 397–415, 2010

[4] A. Chebotko, S. Chang, and S. Lu, "Scientific workflow provenance querying with security views," The Ninth International Conference on Web-Age Information Management, WAIM, pp. 349-356,2008

[5] A. P. Chapman, A. Arbor, and H. V Jagadish, "Efficient provenance storage," 2008.

[6] J. Golbeck and A. Mannes, "Using Trust and Provenance for Content Filtering on the Semantic Web.," MTW, pp. 3–4, 2006.

[7] J. Golbeck, "Combining provenance with trust in social networks for semantic web content filtering," Proven. Annot. Data, 2006.

[8] J. Golbeck and A. Mannes, "Using trust and provenance for content filtering on the semantic web.," MTW, 2006.

[9] S. Wang, A. Padmanabhan, J. D. Myers, W. Tang, and Y. Liu, "Towards provenance-aware geographic information systems," Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst. - GIS '08, p. 1, 2008.

[10] L. Moreau, P. Groth, S. Miles, V. Javier, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The provenance of electronic data ," Communications of the ACM , pp. 1–9, 2007.

[11] R. Angles and C. Gutierrez, "Business process as a service: chances for remote auditing," ACM Comput. Surv., pp. 398–403, Jul. 2008.

[12] R. Angles and C. Gutierrez, "SPADE : Support for provenance auditing in distributed environments," ACM Comput. Surv., 2008.

[13] P. Groth and S. Miles, "A model of process documentation to determine provenance in Mash-ups," vol. V, no. December, pp. 1–30, 2007.

[14] P. T. Groth and L. Moreau, "Recording process documentation for provenance," no. 107, pp. 1–14.

[15] F. Curbera, Y. Doganata, A. Martens, and N. K. Mukhi, "Business provenance – A technology to increase traceability of end-to-end operations," pp. 100–119, 2008.

[16] S. Goedertier and J. Vanthienen, "Designing Compliant business processes with obligations and permissions," pp. 5–14, 2006.

[17] M. P. Papazoglou, "Making business processes compliant to standards and regulations," 2011 IEEE 15th Int. Enterp. Distrib. Object Comput. Conf., pp. 3–13, Aug. 2011.

[18] L. T. Ly, S. Rinderle-ma, D. Knuplesch, and P. Dadam, "Monitoring business process compliance using compliance rule graphs." OTM Conferences ,pp.82-99, 2011

[19] J. Park, D. Nguyen, and R. Sandhu, "A provenance-based access control model," 2012 Tenth Annu. Int. Conf. Privacy, Secur. Trust, pp. 137–144, Jul. 2012.

[20] E. Barka and R. Sandhu, "A role-based delegation model and some extensions," 23rd Natl. Inf. Syst. Secur. Conf., 2000.

[21] I. Technology and A. Dean, "Framework for role-based delegation models," ACSAC, 2000

[22] P. Bammigatti and P. Rao, "Delegation in role based access control model for workflow systems," Int. J. Comput. Sci. Secur., no. 2, pp. 1–10, 2008.

[23] E. Barka and R. Sandhu, "Role-based delegation model/ hierarchical roles (RBDM1)," 20th Annu. Comput. Secur. Appl. Conf., pp. 396–404, 2004.

[24] J. Crampton and H. Khambhammettu, "Delegation in role-based access control," Int. J. Inf. Secur., vol. 7, no. 2, pp. 123–136, Sep. 2007.

[25] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong, "Access control in collaborative systems," ACM Comput. Surv., vol. 37, no. 1, pp. 29–41, Mar. 2005.

[26] S. Schefer, M. Strembeck, J. Mendling, and A. Baumgrass, "Detecting and resolving conflicts of mutual-exclusion and binding constraints in a business process context," OTM 2011, October, 2011.

[27] D. Nguyen, J. Park, and R. Sandhu, "Dependency path patterns as the foundation of access control in provenance-aware Systems," TaPP'12 Proc. 4th USENIX Conf. Theory Pract. Proven., 2012.

[28] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," 15th Natl. Comput. Secur. Conf., pp. 554–563, 1992.

[29] M. Moniruzzaman and K. Barker, "Delegation of access rights in a privacy preserving access control model," PConference Privacy, Secur. Trust, pp. 124–133, 2011.

[30] C. Ye and Z. Wu, "An attribute-based delegation model and its extension," J. Res. Pract. Inf. Technol., vol. 38, no. 1, pp. 3–17, 2006.

[31] K. Gaaloul, A. Schaad, and U. Flegel, "A secure task delegation model for workflows," Second Int. Conf. Emerg. Secur. Information, Syst. Technol. Secur., pp. 10–15, 2008.

[32] L. Moreau, "The foundations for provenance on the web," Found. Trends Web Sci., pp. 99–241, 2010.

[33] E. Barka and R. Sandhu, "Framework for agent-based role delegation," IEEE Int. Conf. Commun., pp. 1361–1367, 2007.

[34] H. Hsu and F. Wang, "A Delegation framework for task-role based access control in WFMS.," J. Inf. Sci. Eng., vol. 1028, pp. 1011–1028, 2011.

[35] S. L. Osborn and H. Wang, "A survey of delegation from an RBAC perspective," J. Softw., vol. 8, no. 2, pp. 266–275, Feb. 2013.

[36] D. Nguyen, J. Park, and R. Sandhu, "A provenance-based access control model for dynamic separation of duties," 2013 Elev. Annu. Conf. Privacy, Secur. Trust, pp. 247–256, Jul. 2013.