Policy Engineering in RBAC and ABAC

Saptarshi Das[1], Barsha Mitra[2], Vijayalakshmi Atluri[3(✉)], Jaideep Vaidya[3], and Shamik Sural[1]

[1] Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur, India
saptarshidas13@iitkgp.ac.in, shamik@cse.iitkgp.ernet.in
[2] Department of CSIS, BITS Pilani Hyderabad Campus, Hyderabad, India
barsha.mitra@hyderabad.bits-pilani.ac.in
[3] MSIS Department, Rutgers University, Newark, USA
atluri@rutgers.edu, jsvaidya@business.rutgers.edu

Abstract. Role-based Access Control (RBAC) and Attribute-based access control (ABAC) are the most widely used access control models for mediating controlled access to resources in organizations. In RBAC, permissions are associated with roles, and users are assigned to appropriate roles. Therefore, it is imperative that a proper set of roles is necessary for the efficient deployment of RBAC. Most organizations possess a set of existing user-permission assignments which can be used to create appropriate roles. This process, known as role mining, is an important and challenging task in the deployment of RBAC in any organization. On the other hand, in ABAC, the access decisions depend on the attributes of the various entities and a set of authorization rules (policies). The efficiency of an ABAC model relies upon the strength and correctness of the authorization rules. Similar to role mining in RBAC, the process of constructing an appropriate set of ABAC authorization rules, known as policy engineering, is crucial for the implementation of ABAC. Regardless of the differences in RBAC and ABAC, the problems of role mining in RBAC and policy engineering in ABAC are quite similar and equally important for the corresponding access control models. In this chapter, we explore the role mining problem and the policy engineering problem along with their existing solution strategies and identify future directions of research in these two areas.

Keywords: Role-Based Access Control (RBAC) · Role mining
Attribute-Based Access Control (ABAC) · Policy engineering
Top-down · Bottom-up · Constraints

1 Introduction

The workflow of any organization depends on the continuous and consistent execution of the assigned tasks by all the employees belonging to that organization. The execution of these tasks, in turn, requires that each and every employee

P. Samarati et al. (Eds.): Jajodia Festschrift, LNCS 11170, pp. 24–54, 2018.
https://doi.org/10.1007/978-3-030-04834-1_2

be given the necessary authorizations and privileges. Employees can acquire the relevant permissions based on some predefined rules, policies and mechanisms. These rules, policies and mechanisms need to ensure not only that each user is given all the required permissions but also that no user is given any extra privilege. Failure to ensure the first aspect may lead to discontent among users or at most, may create some sort of hindrance in the smooth execution of tasks. However, failure to take care of the second aspect will most definitely lead to serious security breaches which can cause far more severe damages than displeasure or discontinuity in organizational workflow. Thus, the rules, policies and mechanisms need to be enforced properly so that none of the above mentioned adverse scenarios occur at any point of time.

Several access control models have been proposed over the past years. Of these, the Role-Based Access Control (RBAC) model [24,74] has become a popular and prominent model since the last decade of the 20th century. Roles are the central elements of the RBAC model. A role is a collection of permissions. Each user is assigned one or more permissions. Hence, in RBAC, users acquire the requisite permissions through their assigned roles. The advantage of RBAC is that it creates an intermediate layer between the users and the permissions thereby, adding a level of stability to the somewhat volatile relationships existing among the users and the permissions. The assignment of permissions to users can vary quite frequently with time, but the membership of a user to a role or the composition of a role is likely to vary infrequently. As a result, RBAC significantly reduces the administrative cost. To successfully implement RBAC, it is necessary to create a set of roles. Role mining is one of the techniques to create roles.

Like any other access control model, RBAC also is not without some drawbacks. RBAC, though being a very appealing choice in case of intra-organizational access control, becomes unsuitable for scenarios where inter-organizational access control is to be considered. The primary reason behind this is that the nature of the roles as well as the permissions present in them may not be uniform across organizations. Thus, the same role will assign different permissions to users in different organizations. In order to cater to the needs of the diverse inter-organizational interactions, the Attribute-Based Access Control (ABAC) model [34,36,41] was proposed.

Attribute-Based Access Control (ABAC) [35] is rapidly emerging as the desired access control model for providing restricted access to organizational resources and to cater to the needs of inter-organizational access control. This model was proposed as a general model which offers all the benefits of the existing access control models, like Discretionary Access Control (DAC) [54], Mandatory Access Control (MAC) [73], and Role-Based Access Control (RBAC) [74]. ABAC mediates access based on the attributes of the requesting user, the requested objects and the environment in which the request is made. ABAC essentially depends on defining a policy consisting of many rules, which are evaluated for deciding access to resources. Thus, for effective working of ABAC, an appropriate set of rules is required to be created. Since a majority of the organizations already have a set of accesses which represent the resources accessible by each

user, this information can be capitalized to form a set of rules. Also, rules can be constructed by careful evaluation of the different business processes of the organization. This process, known as policy engineering, is a major challenging task in the overall process of implementing ABAC in any organization. In recent years, a number of policy engineering methods have been developed, which consider basic components as well as the different features of the ABAC model.

In this chapter, we focus on the two above mentioned access control models. We shall outline some preliminaries related to the models as well as discuss several aspects regarding the policy engineering work in these two models. Specifically, Sect. 2 discusses overview of RBAC, and the different role engineering techniques. In Sect. 3, we first present certain preliminaries related to the ABAC model followed by a detailed discussion of the different ABAC policy engineering techniques. Finally, Sect. 4 concludes the chapter.

2 Policy Engineering in Role-Based Access Control (RBAC)

In this section, we first present a brief overview of the RBAC model in Subsect. 2.1. This is followed by a discussion on role engineering and role mining in Sub-sects. 2.2 and 2.3 respectively. Sub-sects. 2.4 and 2.5 focus on the different unconstrained and constrained variants of the role mining problem respectively. Future directions of research in role engineering and role mining are highlighted in Sub-sect. 2.6.

2.1 Overview of the Model

In this sub-section, we discuss the basic concepts related to the RBAC model. The components that constitute the model are as follows [74]:

- a set of users U
- a set of roles R
- a set of sessions S
- a set of objects OBJS
- a set of operations OPS
- a permission set P such that each member of P is a tuple (op, obj) such that $op \in$ OPS and $obj \in$ OBJS
- a user-role assignment relation UA representing the individual role assignments of each user. UA $\subseteq$ U $\times$ R
- a function $assigned\_users$: R $\rightarrow 2^{\text{U}}$, the mapping of the set R onto the powerset of U. This function is used to derive the set of users to whom a particular role has been assigned. Thus, $assigned\_users(r) = \{u \mid (u, r) \in$ UA$\}$
- a role-permission assignment relation PA depicting the composition of each of the roles in terms of their constituent permissions. PA $\subseteq$ R $\times$ P

- a function $assigned\_permissions : R \rightarrow 2^P$, the mapping of the set R onto the powerset of P. This function is used to determine the permissions included in a specific role. Thus, $assigned\_permissions(r) = \{p \mid (r, p) \in PA\}$
- a partial order called role hierarchy RH which is a subset of $R \times R$. RH captures the relationships among the senior and the junior roles
- a collection of several semantic constraints like mutually exclusive roles, cardinality constraints, etc.

The operations that can be carried out on the objects are represented in the form of the abstractions known as permissions. The set of roles assigned to each user is captured in the user-role assignment relationship UA and the permission set included in each role is depicted using the role-permission assignment relationship PA. RBAC is not a linear monolithic model. Therefore, relationships exist not only among users and roles and roles and permissions, but also among the roles themselves, thereby creating a hierarchy among the roles. A natural extension of this hierarchy is the notion of senior and junior roles. The role hierarchy seamlessly captures the hierarchical structure existing in any organization. The membership of a user to a senior role implies his/her implicit assignment to the related junior roles as well as the acquisition of the permissions included in each of the junior roles. The constraints present in RBAC adds a semantic flavor to it. Constraints reflect several organizational aspects which may or may not relate directly to the security aspect of the model. Mutually exclusive roles ensure that a single user is never allowed to perform all the tasks related to a sensitive job. Cardinality constraints like the highest number of roles that can be assigned to a user or the maximum number of users to whom a particular role can be assigned balance the workload among the different users whereas constraints like the maximum number of permissions permissible per role and the number of roles in which a permission can be present help to make sure that the permission distribution across the roles is uniform.

In order to successfully and effectively implement RBAC, any organization requires to come up with a set of roles. These roles should capture all the permission assignments of the users as well as specific organizational needs. Role engineering is the process of creating the required set of roles [4,11,18,76]. The major cost of deploying RBAC involves the process of role engineering according to a NIST report [69]. We discuss role engineering in the next sub-section.

2.2 Role Engineering

Role engineering plays a pivotal role in the successful deployment of RBAC. In order to implement the RBAC model, a set of roles is required which ensures that all the users possess the relevant permissions to execute their designated tasks. Also, it needs to be ensured that only these permissions are made available to the users. Any fault in the role creation process may either cause some hindrances for some users when they try to access certain resources or may result in unauthorized accesses. All kinds of errors in the role generation process that lead to the second scenario should be removed completely in order to ensure the

proper functioning of the system. In addition to creating a set of roles, role engineering can also take into account several constraints and determine a hierarchy among the roles. Role engineering can be broadly categorized into two types - (i) top-down [68,70] and (ii) bottom-up [21,52,82]. We next discuss each of these two approaches to role engineering in detail.

Top-Down: Top-down role engineering approach begins by analyzing the structure of the organization to identify the business processes that constitute its workflow. On deeper analysis, these business processes are found to be composed of job functions each of which in turn binds together a specific number of tasks. A certain set of permissions is required to carry out each task successfully. Once the permissions necessary for carrying out the tasks are identified, these permissions are put together to create the individual roles. Thus, in the top-down approach, starting from the top-level organizational structure, the business processes and job functions are repeatedly decomposed to find out the lowest level of granularity of access control, i.e., the permissions for determining the role set.

This methodology of role creation was first introduced by Coyne [18]. Subsequently, several others also put forth processes of role creation that correlated organizational theory with RBAC concepts [19] or was based on UML concepts [22,23,76]. Kern et al. [43] amalgamated the concept of role life cycle with role engineering. Other top-down role engineering approaches that have been proposed include process-oriented role engineering [70] and scenario-driven role engineering [4,68,78].

The top-down role engineering approach fails to take into account the existing permission assignments of the users of the organization and may end up creating roles which require changes to be made in these assignments. The consequent revocation and re-assignment of roles may create a sense of apprehension or even aversion among the employees and may ultimately hamper the smooth working of the organization. Moreover, the top-down approach requires a massive amount of human effort and hence is prone to intentional or unintentional errors. Also, since human effort is involved in top-down role engineering, it is not a scalable approach when hundreds or thousands of business processes, users and permissions are present. However, efforts have been made to automate top-down role engineering [67] so as to eliminate the human factor from this method.

Bottom-Up: Bottom-up role engineering was proposed as an alternative to top-down role engineering so that the former did not suffer from the drawbacks of the latter approach. Role mining [21,25,52,79,82] is a bottom-up technique of role engineering. Role mining starts at the permission level by considering the existing permission assignments of the users of the organization. The permission assignment information of the users is represented using a user-permission assignment or UPA relation. The UPA is a many-to-many relation since each user can be assigned more than one permission and each permission can be made available to more than one user. Role mining takes as input the UPA and produces two many-to-many relations - one is the user-role assignment (UA)

relation and the other is the role-permission assignment (PA) relation. Being an algorithmic approach, role mining can be easily automated, thereby completely eliminating the issues related to scalability and any kind of human error. Due to these reasons, role mining has become quite popular and has gained wider spectrum of acceptability than the top-down role engineering techniques.

Inspite of having several advantages, role mining is not without drawbacks. Since role mining takes into account only the permission assignment of the users and leaves out analyzing the business processes of the organization, role mining may create roles that may not directly correlate to the business processes and consequently the job functions of the organization. To remove this drawback as well as consolidate the benefits of the top-down and bottom-up techniques, *hybrid* role engineering approaches [26,27,63] have also been proposed. The hybrid approach not only ensures that the role generation process is scalable, automated and free of human errors, but also helps to create semantically meaningful roles by incorporating the information related to the business process into role creation.

2.3 Role Mining

Role mining, a bottom-up role engineering approach involves creation of a set of roles and the appropriate assignment of these roles to users from the input UPA. The UPA can be represented as a boolean matrix where users correspond to rows and columns correspond to permissions. The assignment of a permission to a user is depicted by putting a 1 in the corresponding cell of the UPA. The output of role mining consists of the UA and the PA relations. The UA and the PA can be represented as boolean matrices. Each row of the UA corresponds to a user and each column corresponds to a role. If a role r is assigned to a user u, then the entry (u, r) of the UA matrix is set to 1. The rows of the PA matrix correspond to roles and the columns correspond to permissions. The inclusion of a permission in a role is indicated by setting the corresponding entry of the PA to 1. Thus, role mining is a boolean matrix decomposition approach in which two boolean matrices, the UA and the PA are obtained by decomposing a single boolean matrix, the UPA. The output UA and PA can be combined together to get the input UPA. Thus, UA $\otimes$ PA = UPA, where $\otimes$ is the boolean matrix multiplication operator. Role mining may also sometimes additionally create the role hierarchy.

While any arbitrary but correct set of roles may be generated from the UPA, often, the objective is to create a minimal set of roles. In this context, a minimal set of roles is one that is optimal with respect to some role mining metric. The problem of generating an optimal role set from the input UPA is termed as the *Role Mining Problem* (RMP). The variant of the role mining problem that considers optimality as the number of roles is Basic-RMP. The formal definition of Basic-RMP as defined by Vaidya et al. [80] is given below.

Definition 1. *Basic-RMP*

Given a UPA, create a set of roles R, a UA and a PA such that |R| is minimized and the output is consistent with the UPA (|R| = number of roles in R).

The output of Basic-RMP is said to be consistent with the input UPA if the user-permission assignment relation obtained by combining the UA and the PA is same as the UPA.

Basic-RMP can be defined using matrix representation notations also. Let us assume that |U| equals m, |P| equals n and |R| is equal to k. Here, |X| represents the size of any relation X. If X can be represented as a Boolean matrix, then |X| is given by the number of 1s present in it. Thus, UA is an $m \times k$ matrix, PA is a $k \times n$ matrix and UPA is an $m \times n$ matrix. Basic-RMP can be stated as: Given an $m \times n$ UPA, create a minimal sized role set R, an $m \times k$ UA and a $k \times n$ PA such that

$$\text{UA} \otimes \text{PA} = \text{UPA} \tag{1}$$

The output of Basic-RMP is said to be consistent with the input UPA if it satisfies Eq. 1. In addition to |R|, several other role mining metrics are also present such as |UA| + |PA| [52], |R| + |UA| + |PA| [89] or a weighted structural complexity (WSC) measure [62,63].

In certain cases, if a certain amount of mismatch is allowed between the input UPA and the user-permission assignments obtained by combining the UA and the PA, then the number of roles can be minimized further. However, the trade-off is a more restrictive RBAC configuration which deprives some users of certain permissions. Also, keeping the target number of roles constant, this amount of mismatch can also be minimized. Apart from these, cardinality constraints and separation of duty constraints [74] can also be considered during role mining. Depending on the chosen minimization criterion, many variants of Basic-RMP such as, δ-approx RMP [80], MinNoise RMP [80], Edge-RMP [52], Weighted Structural Complexity Optimization Problem [63] have been proposed over the past years.

In Sub-sect. 2.4, we focus on the role mining problem variants and approaches that do not consider any constraints and in Sub-sect. 2.5, we present those which take into account several constraints that are part of the RBAC model. Figure 1 shows an overall classification of the different RMP variants, their corresponding optimization metrics and the solution strategies used denoted by the leaf nodes at the bottom.

2.4 Unconstrained Role Mining

In this sub-section, we discuss the RMP variants that do not consider any constraints and only aim at minimizing a specific optimization metric. We refer to these problem variants as unconstrained RMP variants. An optimization metric for role mining is expressed in terms of the sizes of one or more RBAC components. Depending on whether the size of a single RBAC component is considered

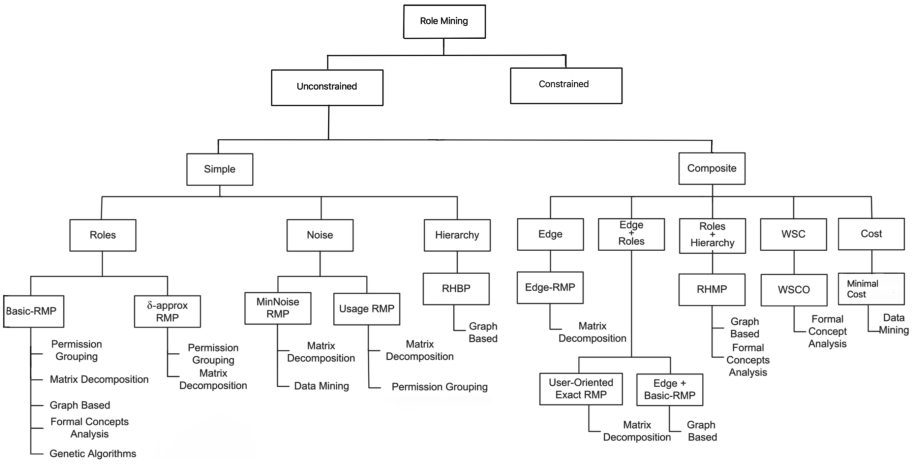

Fig. 1. Role mining classification

or the cumulative sizes of multiple RBAC components are considered, we classify the optimization metrics into two sub-categories - (i) Simple (involving a single RBAC component) and (ii) Composite (involving multiple RBAC components). Simple optimization metrics that exist in the literature include the total number of roles, the deviation of the role mining output from the input UPA calculated as the number of mismatches between the two and the size of the role hierarchy. The Composite category includes metrics such as the cumulative sizes of a combination of the RBAC components like the sizes of the set of roles, the UA relation, the PA relation and the role hierarchy. Most of these unconstrained problem variants have been shown to be NP-hard. We have already presented the formal definition of Basic-RMP. Next, we discuss the other RMP variants.

Simple Optimization Metrics: While the target of Basic-RMP is to come up with the minimum number of roles from an input UPA, several other variants of RMP have been proposed depending on the chosen optimization metric. Each variant aims to minimize the chosen metric such that the solution either exactly reconstructs the UPA or approximates it by allowing a limited degree of mismatch. These problem variants are presented next.

δ-approx RMP: Proposed by Vaidya et al. [80], δ-approx RMP tolerates a pre-specified degree of mismatch between the role mining output and the input UPA. δ-approx RMP can be defined as follows:

Definition 2. *δ-approx RMP*

Given a UPA and a threshold δ, create a role set R, a UA and a PA, such that $||UA \otimes PA - UPA||_1 \leq \delta$ and $|R|$ is minimized.

In the above definition, $\| \cdot \|_1$ represents the L_1 norm and δ denotes the allowed number of mismatches by which the user-permission assignments com-

puted by combining the UA and the PA differ from the UPA. The higher the value of δ, the lower is the number of roles obtained from role mining. However, a high value of δ will make the output RBAC configuration too restrictive. Basic-RMP is a special case of δ-approx RMP where $\delta = 0$.

MinNoise RMP: Instead of pre-defining the number of mismatches and minimizing the number of roles, the complementary approach can also be adopted, i.e., minimizing the number of mismatches keeping the number of roles constant. The RMP variant which does this is referred to as the MinNoise RMP [80]. The number of mismatches between the input user-permission assignments and the ones obtained by combining the output UA and the PA is termed as noise. The input to MinNoise RMP is the UPA and the target number of roles k. The generated output consists of k roles, a UA and a PA such that $||\text{UA} \otimes \text{PA - UPA}||_1$ is minimized. In [80], the authors have mapped MinNoise RMP to the Discrete Basis Problem [56].

Usage RMP: Usage RMP [53] takes as input a set of role-permission assignments apart from the UPA and finds a UA and $||\text{UA} \otimes \text{PA - UPA}||_1$ is minimized. Usage RMP is applicable for organizations where a set of roles already exists. For such organizations, a new role set is not required to be created. Instead, only the roles are appropriately assigned to the users so that the degree of mismatch is minimized. Usage RMP reduces the effort of role mining by limiting the task to creating only the UA.

Role Hierarchy Building Problem: The visual representation of a role hierarchy can be obtained by drawing a directed acyclic graph where roles are represented as nodes and the relationships among senior and junior roles are denoted using edges. A role hierarchy containing the minimum number of edges is said to an optimal role hierarchy.

The Role Hierarchy Building Problem (RHBP), proposed by Guo et al. [29] is an RMP variant which aims to build an optimal role hierarchy given a role set. A role hierarchy is said to be a Complete Role Hierarchy (CRH) if it contains the inheritance relationships between all pairs of roles. The formal definition of RHBP is as follows:

Definition 3. *Role Hierarchy Building Problem*

Given a UPA, a role set R, a UA and a PA, create a complete role hierarchy $RH = G(V, E)$ *where G is the graphical representation of RH, V denotes the set of vertices and E represents the set of edges such that* $|E|$ *is minimal.*

Composite Optimization Metrics: In contrast to the simple role mining metrics, composite role mining metrics consider either a non-weighted or a weighted sum of the sizes of more than one RBAC component. Based on the particular composite metric chosen, different RMP variants exist in the literature. Choosing a composite metric may considerably increase the effort required for role mining. However, composite metrics reduce, to a great extent, the administrators' effort for managing and maintaining the finally deployed RBAC system.

Edge-RMP: Edge-RMP [52,81], a variant of Basic-RMP attempts to reduce redundant roles as well as redundancy in user-role assignments. It fulfills this objective by considering the following minimization criterion - |UA| + |PA|. Edge-RMP also considerably reduces the administrative effort for managing the deployed RBAC configuration.

User-Oriented Exact RMP: The objective of User-Oriented Exact RMP [50,51] is to take the perspective of the end-user into consideration while deriving an RBAC state. An RBAC configuration that does not over burden any user with too many role assignments is more preferable to the users than a configuration which contains a large number of role assignments for each user. Therefore, User-Oriented Exact RMP aims to minimize |R| + |UA|. |UA| can be trivially minimized by making the number of roles and the number of users equal and assigning a single role to each user. However, this kind of a solution contradicts the principal objective of role mining which is to create roles by grouping permissions as well as users. Hence, |R| is also included in the optimization metric. The metric used by User-Oriented Exact RMP is a weighted sum of |R| and |UA|, i.e., $w_r.|\mathrm{R}| + w_u.|\mathrm{UA}|$. In this context, w_r and w_u denote the relative weightage given to the size of the respective RBAC components.

Edge + Basic-RMP: Zhang et al. [89] proposed Edge + Basic-RMP. It aims to minimize |UA| + |PA| + |R|. Edge + Basic-RMP thus reduces the overall administration effort to manage the resulting RBAC state. Consequently, it takes into account both end-user and administrator's perspectives. This RMP variant can minimize the chosen role mining metric even if partial role definitions are available as input apart from the UPA.

Role Hierarchy Mining Problem: The Role Hierarchy Mining Problem (RHMP) [29] was proposed by Guo et al. For this problem, no set of roles exists. Therefore, solving this RMP variant requires creating the role hierarchy along with deriving a role set. The objective here is to minimize the total number of roles as well as the size of the role hierarchy. The formal definition of RHMP is presented below:

Definition 4. *Role Hierarchy Mining Problem*

Given a UPA, the objective is to create a role set R, UA, PA and a complete role hierarchy RH = G(V, E) such that RH is consistent with UPA and |R| + |E| is minimal.

Since RHMP aims to find a minimal set of roles and then create an optimal hierarchy from this role set, the sizes of both R and RH are included in the minimization criterion.

Weighted Structural Complexity Optimization (WSCO) Problem: The metric Weighted Structural Complexity (WSC) was introduced by Molloy et al. [63]. WSC is expressed as a weighted sum of |R|, |UA|, |PA| and |RH|. Additonally, WSC also considers a direct user-permission assignment (DUPA) relation, in case it is available. DUPA consists of the isolated user-permission assignments which cannot be included in a role. Let the weights associated with each of R,

UA, PA, RH and DUPA be w_1, w_2, w_3, w_4, and w_5 respectively, each of which is a non-negative rational number. WSC is calculated as: $w_1.|\text{R}| + w_2.|\text{UA}| + w_3.|\text{PA}| + w_4.|tran\_re(\text{RH})| + w_5.|\text{DUPA}|$. Here, $tran\_re(\text{RH})$ gives the minimum sized set containing the relationships which are equivalent to those present in RH. The RMP variant that minimizes WSC is referred to as the Weighted Structural Complexity Optimization (WSCO) problem [63]. WSCO can be considered as a generalized version of all the RMP variants since by appropriately setting the values of the weights, WSCO can be reduced to different RMP variants.

Among all the optimization metrics discussed so far, WSC is the most complex since it tries to minimize the sizes of a number of RBAC components simultaneously. Though apparently this might seem to be a very appealing choice, at times, minimization of the different components might conflict with each other, consequently, resulting in an RBAC state that is not meaningful. The RMP variants presented here can be further categorized as exact and inexact variants depending upon whether the output generated is consistent with the input UPA. Basic-RMP, User-Oriented Exact RMP, Edge-RMP, RHBP, RHMP, WSCO Problem and Edge+Basic RMP are exact variants whereas MinNoise RMP, δ-approx RMP and Usage RMP can be considered as inexact variants.

Cost Based Metric: A cost based metric was proposed by Colantonio et al. [11]. This metric targets to minimize a cost function $f = w_U|\text{UA}| + w_P|\text{PA}| + w_R|R| + w_C \sum_{r \in \text{R}} c(r)$, where each of w_U, w_P, w_R and w_C is greater than or equal to 0. The function f captures the cost of considering business information in the function c separately from the cost incurred by the role set and the costs of the UA and the PA. The problem of creating a minimal cost role set is equivalent to Basic-RMP when $w_R = 1$ and $w_U, w_p, w_C = 0$.

Noise Consideration: In scenarios where there are erroneous assignments or noise present in the input UPA, it is essential to identify and cleanse the noise before creating the RBAC configuration. Otherwise, the mined RBAC configuration will be erroneous as well. Several techniques have been proposed for identification of noise present in the input which include a rank reduced matrix factorization approach proposed by Molloy et al. [64], an association rule mining based algorithm presented by Huang et al. [37], etc.

Solution Strategies: Since the RMP variants are NP-hard problems, a number of heuristic approaches have been adopted to solve them. Permission grouping based strategies include the ones proposed in [7,80,82,83,91], while problem mapping based techniques include [21,38,39,79]. In addition to these, matrix decomposition based approaches [52,53], graph theoretic algorithms [13,15,29,89], formal concepts analysis based techniques [62,63] are also present. Moreover, it has been shown that data mining techniques and genetic algorithms can be used to perform role mining [1,11,71,72,90]. Approaches to mine roles meaningful from a business perspective have been presented in [12,14,16,17,45,55,65,85] and [86]. Recently, a role engineering method has been

proposed which can be used to create RBAC states in large organizations in a scalable manner [20].

Temporal Mining of Roles: Temporal Role-Based Access Control (TRBAC) model [5] is an extension of the RBAC model. In TRBAC, each role has an associated temporal constraint specifying the time duration for which the role is enabled. These roles have been referred to as temporal roles and the process of mining these roles is termed as temporal role mining [59]. The temporal constraints for these roles are specified in a Role Enabling Base (REB). The problem of mining a minimal set of temporal roles has been termed as the Temporal Role Mining Problem (TRMP) [57]. Generalized Temporal Role Mining Problem (GTRMP) [58] is the inexact version of TRMP where a pre-determined number of mismatches is allowed. Another variant of the TRMP is also present in the literature which aims to minimize a metric known as the cumulative overhead of temporal roles and permissions (CO-TRAP) [59], calculated as a weighted sum of |PA| and the size of the REB. The corresponding problem variant is known as the CO-TRAP Minimization Problem (CO-TRAPMP). The role mining algorithms discussed so far are not suitable for mining of temporal roles. Hence, several temporal role mining algorithms have been proposed based on subset enumeration [58], matrix decomposition using many-valued concepts [59] or algorithms which are extensions of the traditional role mining algorithms [60].

2.5 Constrained Role Mining

Several constraints have been incorporated in RBAC like mutually exclusive roles, cardinality constraints and pre-requisite roles. Cardinality constraints correspond to different organizational policies and rules in an RBAC state. The cardinality constraints indicate at most how many roles can be assigned to a user (C_1) or at most how many users can be assigned to a specific role (C_2) or the highest number of permissions to be included in a role (C_3) or the upper bound on the number of roles in which a permission can be present (C_4). In the role mining literature, C_1 has been named as the *role-usage cardinality constraint* and C_4 has been referred to as the *permission-distribution cardinality constraint* [31]. Similarly, C_2 and C_3 respectively can be termed as *role-distribution cardinality constraint* and *permission-usage cardinality constraint*. The output of role mining should be such that the required constraints are satisfied.

The RMP variant proposed in [50,51] considers the role-usage cardinality constraint (C_1) and is an user-oriented role mining problem. It attempts to prevent over burdening of users with too many role assignments. Two versions of the constrained User-Oriented RMP have been presented - (i) Exact version and (ii) Approximate version. As the names suggest, the first one is an exact version while the second one is an inexact version. The respective problem definitions are presented below.

Definition 5. *User-Oriented Exact RMP*

Given a UPA and $t > 0$, find R, a UA and a PA such that $|R|$ is minimum, the solution is consistent with the input UPA, and no user is assigned more than t roles.

Definition 6. *User-Oriented Approximate RMP*

Given a UPA, $t > 0$ and a positive fractional number f, find R, a UA and a PA such that $|R|$ is minimum, the UA and the PA when combined reconstructs the input UPA with an error rate less that is at most f, and no user is assigned more than t roles. (Error rate denotes the fraction of the mismatched UPA entries.)

Two approaches have been presented for solving the above mentioned problem variants. User-Oriented Exact RMP can be solved using the following iterative greedy strategy - Select the candidate role which when assigned to appropriate users covers the maximum number of user-permission assignments till $t - 1$ (i.e., $C_1 = t$) roles have been assigned to each user. After that, the remaining permission assignments of each user are collectively put in a single role and is assigned to the corresponding user. User-Oriented Approximate RMP can also be solved by adopting a similar strategy. The only difference is that the iterative role selection terminates when the upper bound for the allowable degree of mismatches is reached. Other approaches to solve RMP in the presence of the role-usage cardinality constraint include the Role Priority based Approach (RPA) and the Coverage of Permissions based Approach (CPA) proposed by John et al. [42]. RPA first creates a UA and a PA and then enforces the constraint by modifying them whereas CPA enforces the constraint while creating the UA and the PA.

Algorithms to enforce the role-distribution cardinality constraint (C_2) based on the graph theoretic Minimum Biclique Cover [21] based role mining algorithm has been proposed in [32]. The problem variant considering the permission-usage cardinality constraint (C_3) is formally defined in [8], which has been named as the *t-constrained RMP* (i.e., $C_3 = t$). The problem definition is as follows:

Definition 7. *t-constrained RMP*

Given an $m \times n$ UPA and a positive integer $t > 1$, find an $m \times k$ UA and a $k \times n$ PA so that $UA \otimes PA = UPA$ and $\forall\ i,\ 1 \leq i \leq k$, $|\mathcal{PA}_{ij} = 1| \leq t$, where $1 \leq j \leq n$.

The authors have proposed an iterative approach named as t-SMA to solve the t-constrained RMP. Two variants of this algorithm are presented depending upon whether the row containing the least number of permissions is selected (named as t-SMA$_R$) or the column that contains the least number of permissions is selected (named as t-SMA$_C$) in every iteration. Kumar et al. [46] propose a role mining algorithm called as the Constrained Role Miner (CRM) capable of enforcing the permission-usage cardinality constraint. This approach first creates a set of roles by clustering permission sets assigned to a single or multiple users and then enforces the constraint to create the final role set.

Work on handling multiple cardinality constraints have been considered by Harika et al. [31]. The authors propose the Multiple Cardinality Constraint Problem (MCP) which considers both the role-usage cardinality constraint (C_1) and

the permission-distribution cardinality constraint (C_4). The authors show that MCP can be solved by using either the concurrent processing approach or the post-processing approach. The former approach is similar to CPA and the latter is similar to RPA.

In addition to cardinality constraints, the literature on role mining also contains work on enforcing Separation of Duty (SoD) constraints such as the one presented in [75]. The problem variant that has been proposed in this work is referred to as RMP_SoD. The approaches to solve RMP_SoD enforce SoD by determining the corresponding Statically Mutually Exclusive Roles (SMER) constraints. Constraint supported role engineering technique has been proposed in [33] which is capable of enforcing any desired constraint as a post-processing step by modifying an initial RBAC state obtained as the output of a role mining technique. Another constraint satisfaction approach based on satisfiability modulo theories (SMT) solvers is proposed in [40].

Enforcing one or more constraints may lead to the creation of an RBAC configuration of larger size (i.e., the size of one of more components of the constraint satisfied RBAC configuration may be greater than the size of the corresponding component/s in the unconstrained configuration). Nonetheless, these constraints are necessary to reflect different organizational requirements and policies.

2.6 Future Research Directions

The hybrid approach to role engineering combines the advantages of both top-down and bottom-up approaches. The hybrid techniques can be mostly automated but at the same time incorporates some amount of human intervention. Therefore, in these role engineering techniques, the extent of human induced errors is minimized as far as possible and at the same time the limited amount of human intervention helps to create semantically meaningful roles. Though few hybrid techniques have been proposed till date, this can be a promising direction of future research which in turn may further encourage the real-life deployment of these role mining techniques.

Another area of potential research can be attempting to design role mining techniques which can generate semantically meaningful roles as well as make the newly created roles similar to the existing ones as far as possible. Of course, these two objectives need to be properly balanced with the requirement of minimizing the appropriate role mining metric. Also, it is not just sufficient to deploy an RBAC configuration in an organization. Periodic investigation is required to identify obsolete roles and remove them from the system. It would be interesting to look for approaches that can automate this process.

3 Policy Engineering in Attribute-Based Access Control (ABAC)

While RBAC is competent in mediating efficient access control in environments which involve a known set of users, it is relatively ineffective in scenarios involving

sharing of resources among organizations where the total number of users cannot be known *a priori*. Attribute-Based Access Control (ABAC) [35] has recently been proposed to enforce secure access to resources in a dynamic environment. Basically, attributes are characteristics of the subject, the object, and environment conditions. Attributes consist of information in the form of a name-value pair. In ABAC, subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of rules that are specified in terms of those attributes and conditions. In this section, we explore the problem of policy engineering in ABAC. ABAC along with its basic components and the problem of policy engineering in ABAC, together with its different variants and their corresponding solutions are discussed in the succeeding sub-sections.

3.1 Attribute-Based Access Control (ABAC)

In this sub-section and the subsequent sub-sections, first, we give a general overview of the ABAC model and then, we elaborately discuss and classify the basic problem of policy engineering together with its different variants and solution methodologies corresponding to them. Categorization is performed on the basis of the characteristics of the strategies used to construct the rules, the goal of policy engineering, and the mode of solution. Finally, we explore the limitations of existing work and discover new areas of research that can potentially enrich this area of research.

Overview of the Model: ABAC consists of a set of subjects, objects, environmental conditions and a set of access control rules. A subject usually denotes a human or a non-human entity, such as an application or an automated service. An object or resource is an entity that needs to be protected from unauthorized access. An environment defines the context in which an access request is made like *time of day*, *location of access*, etc. In ABAC, attributes are characteristics of the subject, the object, and environment conditions. Attributes consist of information in the form of a name-value pair. Every subject is associated with several attributes, such as *designation*, *experience*, etc., which either individually or in combination, comprises an expression to identify a group of subjects having similar access rights. Similarly, for each object, appropriate values are assigned to a set of object attributes. Typical examples of object attributes include *file type*, *sensitivity level* and *date of creation*. Similarly, examples of environment attributes include *location of access*, *time of access* etc. Access decisions are based on the values of the attributes assigned to the subject, object and environment conditions. A subject requesting to perform operations on an object is granted or denied access based on assigned attribute values of the subject, the object, environment conditions, and a set of rules that are defined in terms of those attribute values and conditions. Each access or access request is represented in the form of a 4-tuple consisting of a subject, an object, an environment condition and an operation. Rules define the access control policy of the organi-

zation. A set of formal notations is given below. We will use the same notations throughout the chapter.

- S: A set of authorized users. Each element of this set is represented as s_i, for $1 \leq i \leq |S|$.
- O: A set of objects which is to be protected. Each element of this set is represented as o_i, for $1 \leq i \leq |O|$.
- E: A set of environmental conditions. Each element of this set is represented as e_i, for $1 \leq i \leq |E|$.
- S_a: A set of subject attributes that can affect access decisions. Each element of this set is represented as sa_i, for $1 \leq i \leq |S_a|$. Each sa_i has a possible set of values it can acquire. Similarly, O_a and E_a represent the sets of object attributes and environment attributes, respectively.
- F_s: $S \times S_a \rightarrow \{k|k$ is a subject attribute value$\}$. The functions F_o and F_e are similarly defined for object and environment, respectively. Essentially, these functions assign values to attributes for all the entities.
- S_v: A set containing the assignment of attributes and their corresponding values for all the subjects. The sets O_v and E_v are defined for object and environment, respectively.
- OP: A set of operations. Each element of this set is represented as op_i, for $1 \leq i \leq |OP|$.
- $\mathfrak{R}$: A set of rules collectively called the ABAC policy. Each member of this set is represented as $\mathfrak{r}_i$, for $1 \leq i \leq |\mathfrak{R}|$.

Each rule $\mathfrak{r} \in \mathfrak{R}$ is a 4-tuple $\langle RS, RO, RE, op\rangle$, where RS, RO and RE represent a conjunction of subject attribute-value pairs, a conjunction of object attribute-value pairs and a conjunction of environment attribute-value pairs, respectively and $\mathfrak{r}[RS]$ represents the subject attribute-value pairs associated with rule $\mathfrak{r}$. $\mathfrak{r}[RO], \mathfrak{r}[RE]$ and $\mathfrak{r}[op]$ are defined similarly. op is the name of an operation. Each attribute-value pair $av \in \{RS \cup RO \cup RE\}$ is an equality of the form $a = c$, where a is the name of an attribute and c is the value associated with a. c is either a constant or a *don't care* represented as "–".

Policy Engineering: One of the most challenging issues in implementing ABAC is to define a complete and appropriate set of rules each of which is known as a policy. This process, known as *policy engineering* [47], has been identified as one of the most difficult and costliest components in implementing ABAC [47]. Similar to that of role engineering, primarily, there are two strategies employed for ABAC policy engineering: top-down and bottom-up. In the top-down approach, rules are constructed by precisely evaluating and breaking down business processes into smaller functionally independent units. These functional units are then associated with accesses from which the rules are constructed. Specifically, this approach defines a particular unit of a business process and then creates rules for it by considering the associated accesses with the job function. However, this approach may ignore some of the existing accesses in the organization. In contrast, the bottom-up approach, also called policy mining

takes into account the existing accesses to construct rules. ABAC policy mining algorithms have been developed to lower the expense of developing an ABAC policy, by partially automating the procedure. However, most organizations have high-level requirement specifications that govern which user, in what conditions, may access what resources. This approach ignores the high-level requirement specifications in organizations that could be very effective for policy engineering. Interestingly, top-down and bottom-up approaches complement each other in terms of their strengths and weaknesses.

Let us consider a scenario where, *Bob* and *Alice* are two entities of an university. Both of them belong to the department of Computer Science and Engineering (CSE). *Bob* is a $faculty$ and *Alice* is a student having roll number 1001. Consider two objects doc_1 and doc_2, both belonging to the CSE department. The types of doc_1 and doc_2 are *questionnaire* and *assignment*, respectively, and *Alice* has the roll number $CS17S1001$. The existing accesses in the university are given in Table 1.

Table 1. Existing accesses in the university

	doc_1	doc_2
Bob	*access*	*access*
Alice	*deny*	*access*

First, we consider the top-down approach where the various departmental authorities and the security officer (SO) identify two independent functional modules in the organization as *prepare question* and *prepare assignment*. The SO allocates doc_1 to *Bob* under the functional module *prepare question*, so that he can prepare the questionnaire for CSE. The rule generated from this assignment can be represented as:
$\langle subject.designation = faculty\ AND\ subject.department = CSE\ AND\ object.type = questionnaire\ AND\ object.department = CSE\rangle$
Similarly, the functional module *prepare assignment* will form the rule:
$\langle subject.designation = faculty\ AND\ subject.department = CSE\ AND\ object.type = assignment\ AND\ object.department = CSE\rangle$

It is to be observed that the formed rules reflect the functional modules of the university but any of the two formed rules doesn't allow *Alice* to access doc_2. Thus, although the rules are meaningful and help understand the functional modules of the university, it ignores an existing access in the university which is undesirable. This is the limitation of using the top-down approach.

In contrast, the bottom-up approach considers the existing accesses in the organization to form the rules. From the given accesses in Table 1, let us form the following rules from the accesses:

$\mathfrak{r}_1 = \langle subject.designation = faculty\ AND\ object.department = CSE\rangle$ and
$\mathfrak{r}_2 = \langle subject.roll\ number = CS17S1001\ AND\ object.type = assignment\rangle$

We see that, rule $\mathfrak{r}_1$ allows *Bob* to access both doc_1 and doc_2. Rule $\mathfrak{r}_1$ can be literally stated as, "allow all faculties to access all objects of department CSE". Similarly, rule $\mathfrak{r}_2$ allows *Alice* to access doc_2 and can be stated as, "subject having roll number $CS17S1001$ is allowed to access objects of type $assignment$". Although the rules $\mathfrak{r}_1$ and $\mathfrak{r}_2$ satisfy the existing accesses in the university, the rules do not reflect the functional modules of the university. Moreover, the rules are not much meaningful. This is the limitation of using the bottom-up approach.

Therefore, an ABAC policy can be constructed either from the functionally independent processes of an organization or a set of existing access data in the organization. From this perspective, the policy engineering problem is a process of constructing a set of authorization rules for an organization from either the natural language policy documents or the set of existing accesses in the organization given that the set of users, the set of resources, the attributes associated with the subjects and objects and their associated values for each subject and object is known.

A trivial solution to the policy engineering problem using the bottom-up model can be formulated by converting each existing access into a separate rule. While such a solution suffices for providing controlled access to the organizational resources, it results in the formation of a large number of rules. Moreover, in case of a new access request, apart from the existing accesses, the rules constructed in this manner will not suffice. Often it is beneficial to fulfill additional constraints such as minimization or maximization of one or more metrics. The problem of specifying an *optimal* set of rules from the set of users, resources, attributes and attribute-value assignments of all the entities is referred as the Policy Engineering Problem (PEP). The fitness of a generated ABAC policy can be represented in terms of the selected measure of optimality. Optimality here may refer to the number of rules constructed, the similarity between the accesses permitted by the constructed ABAC system and the previous system or a Weighted Structural Complexity (WSC). Based on the organizational requirements and the chosen quality metric, different variants of PEP and their corresponding solutions have been proposed in the recent years. Although there are a number of existing policy engineering algorithms, there is no formal classification of the algorithms for policy engineering except broadly categorizing the existing solutions into top-down and bottom-up approaches.

In this chapter, we explore the existing variants of PEP, categorize them, and discuss the proposed solution methodologies. Figure 1 provides the classification of various policy engineering approaches according to the approach and method of solution used. First, we classify PEP on the basis of the approach for solving it i.e., general, top-down and bottom-up approaches which are further classified into different categories based on the metrices and techniques used for solving them. The general approaches for policy mining are categorized into (1) Risk, which associates each access to a potential risk i.e., it quantifies the possible risk or benefit of granting an access. (2) Enumerated, where subjects and objects are assigned a single label for a specific operation and a policy is constructed by enumeration of the subject and object labels. The top-down approaches for

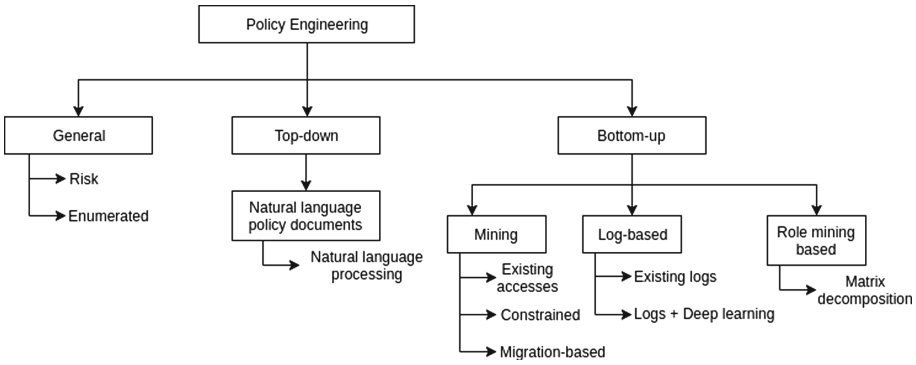

Fig. 2. Classification of policy engineering approaches

policy engineering construct the rules from the high-level descriptions of the business processes available from the natural language policy (NLP) documents available in the organization. The procedures using NLP documents are further categorized into (1) Natural Language Processing, which capitalizes on various natural language processing techniques including point-wise mutual information to identify access control policy sentences within NLP documents. The third category of policy engineering techniques is the bottom-up approach which is further categorized into (1) Mining, which utilizes the existing accesses of an organization to identify a set of rules and can also be performed under various constraints. (2) Log-based, which utilizes the accesses from the logs, then iterates over the accesses extracted from the log to construct rules based on the attributes and their associated values obtained from the entities in the accesses. (3) Role mining based, similar to the role mining problem in RBAC, it first represents the various components of ABAC in a matrix form and mines the attribute-value pairs in the ABAC rules.

Table 2. Different approaches for policy engineering in ABAC

Problem	Input	Output	Minimize	Solution-type
Risk-based [47]	$S, O, OP, S_a, O_a, S_v, O_v, RV$	$\mathfrak{P}$	Risk	Inexact
Enumerated [6]	π_{RBAC}	$\mathfrak{P}$	$WSC(rules)$	Exact
From NLP documents [66]	Sentences from NLP documents	$\mathfrak{P}$	F1-measure	Inexact
Mining [88]	$S, O, OP, S_a, O_a, S_v, O_v$	$\mathfrak{P}$	$WSC(rules)$	Exact
Constrained mining [28]	$S, O, OP, S_a, O_a, S_v, O_v$	$\mathfrak{P}$	$TW(rules)$	Exact
Migration-based [84]	Multiple access control policies	$\mathfrak{P}$	$TotalCost$	Exact
Log [87]	$S, O, OP, S_a, O_a, S_v, O_v, L$	$\mathfrak{P}$	$WSC(rules)$	Inexact
Log + Deep learning [61]	$S, O, OP, S_a, O_a, S_v, O_v, L$	$\mathfrak{P}$	Hamming distance	Inexact
Matrix decomposition [44]	S, O, OP, S_a, O_a, A	$\mathfrak{P}, S_v, O_v$	N.A.	Exact

3.2 Approaches for Policy Engineering

As discussed in Sect. 3.1, the policy engineering problem involves the construction of a set of authorization rules either from the natural language policy documents or from existing accesses in the organization. In this section, we study the various approaches for solving the policy engineering problem for ABAC. Figure 2 shows the general classification of techniques for policy engineering in ABAC. The first subsection describes the general approaches, the second subsection details the top-down approaches and the final subsection focuses on the bottom-up approaches. Table 2 lists the different approaches for policy engineering in ABAC.

General Approaches: The general approaches consist of solutions to PEP which are not based on the high-level functional requirements or the existing accesses of the organization. They are either constructed directly from other traditional access control models or obtained by enumeration. The general approaches are briefly discussed below:

Risk-Based: One of the major concerns while constructing an ABAC policy is the potential risk of allowing an unauthorized access. Risk has been used to assess the efficiency of different RBAC models [3,9]. From this perspective, minimizing the total risk of an ABAC model can be a suitable optimization metric for policy engineering. Krautsevich *et al.* [47] used risk to quantify the possible impairment caused due to unfair use of a granted access. A potential risk value is computed for each possible access. The risk-based policy engineering procedure assumes that permitting an access to a user is associated with the risk that the user may misuse or abuse the obtained access permission. Therefore, the attribute values associated with the rules should be assigned in such a manner that the benefits of granting or denying access minimize the possible risk for the system. The risk-based policy engineering problem is defined below.

Definition 8. *Risk-based PEP*
Given a set of subjects S, a set of objects O, a set of subject attributes S_a, a set of object attributes O_a, attribute value assignments for all subjects S_v, attribute value assignments for all objects O_v, a set of accesses A and a set RV of computed risk values associated with each possible access construct an ABAC policy $\mathfrak{P}$ in such a manner that the total risk calculated from the accesses allowed by $\mathfrak{P}$ is minimum.

The authors do not consider risk for making dynamic access decisions in case of an access request. However, the dynamic access decisions help in constructing balanced ABAC policies in which risk is minimized.

Enumerated: The conventional approach to define ABAC policies is to form logical formulas using the attribute values of the different entities. For instance, ABAC_α [41] and XACML [2] form logical formulas using attribute values. Alternatively, ABAC policies can be specified by enumeration. The Policy Machine

uses enumeration to construct policies. Biswas *et al.* [6] proposed a label-based ABAC model which uses enumeration for constructing ABAC policies. The authors refer to their model as LaBAC. There is one user attribute (uLabel) and one object attribute (oLabel) in LaBAC. An authorization rule in LaBAC corresponding to an access is an enumeration of these two attributes. This makes LaBAC a very basic ABAC model consisting of only one subject attribute and one object attribute.

Top-Down Approach: The top-down approach is like a *clean slate* procedure. Here, a group of authorities in charge of the business processes, with the help of a SO, identifies the functionally independent business processes in the organization and associates them with their corresponding accesses. The authorities and the SO identify the users who perform a specific function and assign them the accesses to the desired objects.

In other words, rules are specified by precisely evaluating and disintegrating business processes into smaller functionally independent units. These functionally independent units are then associated with accesses from which the rules are constructed. Specifically, this approach defines a particular unit of a business process and then creates rules for it by considering the associated accesses with the particular unit. One difficulty of this approach is that it is not always feasible to assemble a team of authorities from multiple departments of the organization within a specified duration to accomplish the objectives of policy engineering. Also, it is human-effort intensive and thus, is prone to errors. Moreover, this approach may ignore some of the existing accesses in the organization.

From Natural Language Policy (NLP) Documents: As it is very difficult to assemble a team of authorities from various departments within a given time period, the existing NLP documents in the organization are sometimes used to identify the different business processes of the organization. Narouei *et al.* [66] present a top-down policy engineering framework for ABAC that employs a deep recurrent neural network to automate the construction of an ABAC policy from unrestricted natural language documents. Majority of organizations have specifications regarding access to organizational resources that state the conditions in which a user can access a particular resource [35]. These documents define security specifications and provide a set of Access Control Policies (ACPs) which contain the permitted accesses. The authors address these documents (high-level requirement specifications) as natural language access control policies (NLACPs) which are specified as statements that regulate and facilitate access to organizational resources. These are expressions in human language that can be transformed to digital policies which mediate machine enforceable access control. The information extracted from NLACPs is used to develop ABAC policies. However, a difficulty in constructing ABAC policies is that the required information to create the authorization rules is usually concealed in the NLACPs, and are hard to identify. This necessitates processing and extracting information from natural language documents. The authors claim their work to be the first attempt to

construct ABAC policies from requirement specification documents and various policy documents which are written in unrestricted natural language.

For evaluation of the obtained results, the authors use recall, precision, and F_1 measure. The portion of ACPs that is relevant is the precision and the fraction of ACPs retrieved correctly is called the recall. For computational purposes, the predictions from the deep neural network classifier are categorized into 4 groups: (1) True positives (TP) corresponding to the correct predictions, (2) True negatives (TN) corresponding to the sentences which are correctly identified as non-ACP sentences, (3) False Positives (FP) representing the sentences incorrectly identified as ACP sentences and (4) False negatives (FN) are the sentences that are identified as non-ACP sentences but are actually not. Precision and recall are calculated as:

$$P = \frac{TP}{TP + FP} \text{ and } R = \frac{TP}{TP + FN}$$

An efficient model will have high values of both precision and recall. The authors express the F_1 measure as the harmonic mean of precision and recall and can be calculated as:

$$F_1 = \frac{2P \times R}{P + R}$$

It may be noted that the value of F_1 tends to shift towards the lower value of precision and recall.

Bottom-Up Approach: The bottom-up approach seeks to capitalize on existing access definitions available in an organization. An organization invests time and effort in defining a set or sets of access control rules and conventions. Rather than using a *clean slate* method, this approach aims to construct authorization rules from these existing accesses. Constructing authorization rules from the existing accesses is called policy mining. ABAC policy mining algorithms have been developed to cut the cost of constructing an ABAC policy by partially automating the process. But, most organizations have specifications in context of different business processes that determine the access decisions regarding organizational resources. This approach ignores the specifications related to the business processes in organizations that have the potential to facilitate the policy engineering process. In other words, the rules formed using the bottom-up approach may fail to reflect the business processes of the organization.

Mining: Xu *et al.* [88] proposed the first known algorithm for mining ABAC policies using a bottom-up approach. Their algorithm constructs an ABAC policy from Access Control Lists (ACLs) and attribute data. The policy mining problem is defined as follows.

Definition 9. *Policy mining problem*
Given a set of subjects S, a set of objects O, a set of subject attributes S_a, a set of object attributes O_a and a set of accesses A, two sets S_v and O_v, which contain all the subjects and objects with their associated attributes and their corresponding

values, respectively and a set of existing accesses A, find an ABAC policy $\mathfrak{P}$ *such that the WSC of* $\mathfrak{P}$ *is minimum.*

Mining can also be used to derive ABAC policy from an RBAC policy and attribute data by converting the RBAC policy into ACLs and converting a role into an attribute and then applying the mining algorithm. The policy mining algorithm works as follows. It iterates over the accesses contained in the given ACL, selects specific accesses and uses them to construct candidate rules, then the candidate rules are generalized to cover additional accesses in the given ACL by substituting conjuncts in attribute expressions with constraints. When the complete ACL has been covered by the constructed candidate rules, the algorithm merges and simplifies the candidate rules to improve the policy. Finally, the algorithm selects the highest-quality candidate rules which are added to the generated policy. The quality metric used in the policy mining algorithm is the WSC metric which is a generalization of the policy size. The WSC of an ABAC rule is a weighted sum of the number of elements of each ABAC component that is present in the rule. Similarly, the sum of the WSCs of the rules of an ABAC policy gives the total WSC of the policy.

Constrained Mining: Policy mining is an effective means for constructing an ABAC policy. However, rules consisting of numerous attributes affect the time required to evaluate each rule in case of an actual access request. Therefore, imposing a constraint on the number of attributes in each rule, along with minimizing the number of attributes in the total policy is beneficial. Gautam *et al.* [28] gave a constrained policy mining algorithm which takes as input an Access Control Matrix (ACM) and constructs a minimal set of ABAC authorization rules in such a way that each rule can have at most a fixed number of attributes. Minimality here refers to the total weight of all the rules. The authors refer to the problem as Constrained ABAC Policy Mining Problem (CAPM) and define the problem as follows.

Definition 10. *Constrained policy mining*
Given an access control matrix A, a set of subject attributes S_a*, object attributes* O_a*, attribute value assignments for all subjects* S_v*, attribute value assignments for all objects* O_v*, and a constant c, construct an ABAC policy* $\mathfrak{P}$ *in such a way that the rules in* $\mathfrak{P}$ *cover all the accesses in A, there are no extraneous accesses permitted by* $\mathfrak{P}$ *which is not present in A and the number of attributes in each rule in* $\mathfrak{P}$ *is at most c and the total weight of the policy i.e.,* $TW(\mathfrak{P})$ *is minimum.*

Here, $TW(\mathfrak{P})$ denotes the total number of attributes in the policy. For a policy consisting of n rules, the total weight of $\mathfrak{P}$ can be denoted as:

$$TW(\mathfrak{P}) = \sum_{i=1}^{n} TW(\mathfrak{r}_i)$$

Migration-Based: The process of upgrading from a traditional access control to a recent access control model is known as policy migration. Many organizations

want to migrate to ABAC for the increased flexibility it offers in regulating controlled access to organizational resources. Any organization migrating to ABAC requires an ABAC policy. Moreover, the need for resource sharing among different organizations necessitates the development of a common policy among them. Quantifying the similarity among different access control policies is the key to constructing a common policy. Lin *et al.* [48,49] present a metric for measuring the similarity between two policies. In this context, Vaidya *et al.* [84] present a framework for migrating to ABAC. Their work is based on a change detection approach that is used to evaluate similarities between security policies of similar or distinct access control semantics. Given a set of policies, they find a common organizational policy with the lowest cost of migration. The cost of migration is calculated on the basis of the changes that occurred from given policies to formed common policy. The change between the policies is identified using the XyDiff tool [10]. The authors mine the policies from access control lists and attribute data. They also provide an extension of the algorithm to detect over-assignment and under-assignment of accesses to a user.

From Logs: We have seen that existing accesses can be used for mining an effective ABAC policy. Alternatively, operation logs can be treated as effective sources of information on existing organizational accesses. Xu *et al.* [87] present the first known algorithm for mining ABAC policies from logs and attribute data. The authors represent a log entry as a 4-tuple, e.g., a log entry is represented as $\langle s, o, op, t\rangle$ where s, o, op and t correspond to a subject, an object, an operation and a time-stamp, respectively. A log record is a collection of such log entries. The problem of mining policies from logs is defined as follows.

Definition 11. *Policy mining from logs*
Given a set of subjects S, a set of objects O, a set of operations OP a set of subject attributes S_a, a set of object attributes O_a, a set S_v of subject attribute data, a set O_v of object attribute data and a log record L, construct a set of ABAC rules $\mathfrak{P}$ such that the WSC of $\mathfrak{P}$ is minimized.

The algorithm works as follows: First, it extracts the accesses from the logs, then it iterates over the extracted accesses, uses selected accesses as bases for forming candidate rules. Then the candidate rules are converted into more generalized rules by replacing some of the attribute expressions with constraints. Generalization of candidate rules results in the coverage of more accesses. Candidate rules are constructed until all the accesses are covered. Finally, the candidate rules are simplified and merged in order to make the policy more efficient. The highest-quality candidate rules are included in the generated policy.

From Logs Using Deep Learning: Iterating over the extracted accesses from the operation logs of an organization is one way of defining an ABAC policy. Alternatively, machine learning techniques can be employed for mining authorization rules from log records. Mocanu *et al.* [61] employ a deep learning technique to interpret rules from logs. Unlike the approach presented in [87], this approach considers the denied accesses along with the permitted accesses. Moreover,

it also considers the issues of under-assignment i.e., the logs may contain some false positive instances like an unauthorized access being permitted, and situations of over-assignment where certain accesses, although authorized, presently do not exist in the log records. The problem definition is similar to the one given in [87]. The authors use Restricted Boltzmann Machines (RBMs) [77] to infer authorization rules from the log records. After training with the log records, the RBM is used to construct the generalized candidate rules. Hamming distance [30] is used to evaluate the quality of the generated policy which measures the reconstruction error.

Matrix Decomposition: Matrix decomposition can also be used to formulate and solve the PEP in ABAC. Krautsevich *et al.* [44], for the first time formalized ABAC in a matrix form and formulated the problem of policy engineering in ABAC. The authors propose the most general policy engineering problem and leave any potential algorithmic solution or quality metric for future work. This method takes as input a set of subjects S, a set of objects O, a set of subject attributes S_a, a set of object attributes O_a and a set of accesses A and produces two matrices S_v and O_v, which contain all the subjects and objects with their associated attributes and their corresponding values, respectively and also represents the rules in an ABAC policy in a matrix form.

3.3 Future Directions

As discussed in the previous sections, both the top-down and bottom-up approaches have their corresponding shortcomings. In order to address the issues faced by the existing algorithms for policy engineering, it is essential to develop methods which can benefit from the advantages of both the approaches. We refer to such methods as the hybrid approaches.

The hybrid approach seeks to utilize both the top-down and bottom-up approaches. Accesses can be gathered using bottom-up methods and evaluated to prevent any unauthorized access. Organizational authorities with the help of SO then can consider the obtained accesses while performing the top-down approach, potentially saving time and effort.

Some organizations often involve multiple business processes with tens of thousands of employees and even more number of resources. In such a scenario, often it becomes very difficult for various authorities from different departments within the organization to understand the business processes of one another and construct an ABAC policy. Therefore, depending exclusively on a top-down approach is not reasonable in the majority of scenarios. Besides, such an organization is likely to have millions of possible accesses, all of which are required to mine a meaningful ABAC policy. It is imperative that obtaining all the accesses is difficult in practice. Conversely, it is easier for the security officer (SO) of the organization to answer in *yes* or *no* when asked whether a given subject can perform a given operation on a given resource in an environment condition.

In such situations, a hybrid approach may prove to be beneficial. The SO can be consulted whether a few accesses pertaining to a certain business process in the organization are allowed or not. This is similar to the top-down

approach. Rules can be inferred from the decisions obtained from the SO in a bottom-up fashion. Thus, this may eventually resolve the issue of leaving out existing accesses in case of top-down approaches. Moreover, as the SO is consulted for accesses related to similar business processes, the rules formed using the bottom-up fashion will be relevant to the business processes of the organization. Therefore, the issue of forming irrelevant rules using the bottom-up approach will also be resolved.

4 Conclusions

In this chapter, we have reviewed policy engineering in the two most widely used access control models - the RBAC model and the ABAC model. Role engineering is a crucial step in the deployment of RBAC. We have discussed the different role engineering techniques present in the current literature. More specifically, we have concentrated on role mining, a bottom-up role engineering approach. We have also discussed the different role mining problem variants and have presented a detailed overview of the different role mining algorithms.

The second half of the chapter discusses the different approaches for policy engineering for ABAC which are essential for the efficient deployment of ABAC in any organization. The existing variants of the policy engineering problem in literature have also been discussed. For both role mining and ABAC policy engineering, we have given a classification of the problem variants and solution strategies based on different criteria. Future directions of research for both role and ABAC policy engineering have also been highlighted in the chapter.

Acknowledgements. Research reported in this publication was supported by the National Institutes of Health under award R01GM118574. The work is also supported in part by the National Science Foundation under grant CNS-1624503. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Data Bases (VLDB), pp. 487–499, September 1994
2. Moses, T., et al.: Extensible access control markup language (XACML) version 2.0. Oasis Standard (2005)
3. Aziz, B., Foley, S.N., Herbert, J., Swart, G.: Reconfiguring role based access control policies using risk semantics. J. High Speed Netw. **15**(3), 261–273 (2006)
4. Baumgrass, A., Strembeck, M., Rinderle-Ma, S.: Deriving role engineering artifacts from business processes and scenario models. In: Proceedings of 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 11–20, June 2011
5. Bertino, E., Bonatti, P.A., Ferrari, E.: TRBAC: a temporal role-based access control model. ACM Trans. Inf. Syst. Secur. (TISSEC) **4**(3), 191–233 (2001)

6. Biswas, P., Sandhu, R., Krishnan, R.: Label-based access control: an ABAC model with enumerated authorization policy. In: Conference on Data and Applications Security and Privacy, pp. 1–12 (2016)
7. Blundo, C., Cimato, S.: A simple role mining algorithm. In: Proceedings of 25th ACM Symposium on Applied Computing (SAC), pp. 1958–1962, March 2010
8. Blundo, C., Cimato, S.: Constrained role mining. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 289–304. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38004-4_19
9. Chen, L., Crampton, J.: Risk-aware role-based access control. In: Meadows, C., Fernandez-Gago, C. (eds.) STM 2011. LNCS, vol. 7170, pp. 140–156. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29963-6_11
10. Cobena, G., Abiteboul, S., Marian, A.: Detecting changes in xml documents. In: International Conference on Data Engineering (IDCE), pp. 41–52 (2002)
11. Colantonio, A., Pietro, R.D., Ocello, A.: A cost-driven approach to role engineering. In: Proceedings of 23rd ACM Symposium on Applied Computing (SAC), pp. 2129–2136, March 2008
12. Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V.: A formal framework to elicit roles with business meaning in RBAC systems. In: Proceedings of 14th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 85–94, June 2009
13. Colantonio, A., Di Pietro, R., Ocello, A., Verde, N.V.: Mining stable roles in RBAC. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IAICT, vol. 297, pp. 259–269. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01244-0_23
14. Colantonio, A., Di Pietro, R., Ocello, A., Verde, N.V.: Mining business-relevant RBAC states through decomposition. In: Rannenberg, K., Varadharajan, V., Weber, C. (eds.) SEC 2010. IAICT, vol. 330, pp. 19–30. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15257-3_3
15. Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V.: Taming role mining complexity in RBAC. Comput. Secur. **29**(5), 548–564 (2010). Special Issue on Challenges for Security and Privacy and Trust
16. Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V.: A new role mining framework to elicit business roles and to mitigate enterprise risk. Decis. Support Syst. (DSS) **50**(4), 715–731 (2011)
17. Colantonio, A., Pietro, R.D., Verde, N.V.: A business-driven decomposition methodology for role mining. Comput. Secur. (COSE) **31**(7), 844–855 (2012)
18. Coyne, E.J.: Role engineering. In: Proceedings of 1st ACM Workshop on Role-Based Access Control (RBAC), pp. 15–16, November 1995
19. Crook, R., Ince, D., Nuseibeh, B.: Towards an analytical role modelling framework for security requirements. In: Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ), pp. 9–10, September 2002
20. Elliott, A., Knight, S.: Start here: engineering scalable access control systems. In: Proceedings of 21st ACM on Symposium on Access Control Models and Technologies (SACMAT), pp. 113–124, June 2016
21. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: Proceedings of 13th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 1–10, June 2008
22. Epstein, P., Sandhu, R.: Towards a UML based approach to role engineering. In: Proceedings of 4th ACM Workshop on Role-Based Access Control, pp. 135–143, October 1999

23. Fernandez, E.B., Hawkins, J.C.: Determining role rights from use cases. In: Proceedings of 2nd ACM Workshop on Role-based Access Control (RBAC), pp. 121–125, November 1997
24. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. (TISSEC) **4**(3), 224–274 (2001)
25. Frank, M., Buhmann, J.M., Basin, D.: Role mining with probabilistic models. ACM Trans. Inf. Syst. Secur. (TISSEC) **15**(4), 1–28 (2013)
26. Frank, M., Streich, A.P., Basin, D., Buhmann, J.M.: A probabilistic approach to hybrid role mining. In: Proceedings of 16th ACM Conference on Computer and Communications Security (CCS), pp. 101–111, November 2009
27. Fuchs, L., Pernul, G.: HyDRo – hybrid development of roles. In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 287–302. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89862-7_24
28. Gautam, M., Jha, S., Sural, S., Vaidya, J., Atluri, V.: Constrained policy mining in attribute based access control. In: ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 121–123 (2017)
29. Guo, Q., Vaidya, J., Atluri, V.: The role hierarchy mining problem: discovery of optimal role hierarchies. In: Proceedings of 24th Annual Computer Security Applications Conference (ACSAC), pp. 237–246, December 2008
30. Hamming, R.: Error detecting and error correcting codes. Bell Syst. Tech. J. **26**(2), 14–160 (1950)
31. Harika, P., Nagajyothi, M., John, J.C., Sural, S., Vaidya, J., Atluri, V.: Meeting cardinality constraints in role mining. IEEE Trans. Dependable Secur. Comput. (TDSC) **12**(1), 71–84 (2015)
32. Hingankar, M., Sural, S.: Towards role mining with restricted user-role assignment. In: Proceedings of 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), pp. 1–5, February 2011
33. Hu, J., Khan, K.M., Bai, Y., Zhang, Y.: Constraint-enhanced role engineering via answer set programming. In: Proceedings of 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS), pp. 73–74, May 2012
34. Hu, V.C., et al.: Guide to Attribute-Based Access Control (ABAC) definition and considerations. Technical report, NIST Special Publication 800-162, January 2014. http://nvlpubs.nist.gov/nistpubs/-specialpublications/NIST.sp.800-162.pdf
35. Hu, V.C., et al.: Guide to attribute based access control (ABAC) definition and considerations. National Institute of Standards and Technology Special Publication (2014)
36. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Attribute-based access control. Computer (IEEE) **48**(2), 85–88 (2015)
37. Huang, C., Sun, J., Wang, X., Si, Y., Wu, D.: Preprocessing the noise in legacy user permission assignment data for role mining - an industrial practice. In: Proceedings of 25th IEEE International Conference on Software Maintenance (ICSM), pp. 403–406, September 2009
38. Huang, H., Shang, F., Liu, J., Du, H.: Handling least privilege problem and role mining in RBAC. J. Comb. Optim. **30**(1), 63–86 (2015)
39. Huang, H., Shang, F., Zhang, J.: Approximation algorithms for minimizing the number of roles and administrative assignments in RBAC. In: Proceedings of 36th Annual IEEE Computer Software and Applications Conference Workshops (COMPSAC), pp. 427–432, July 2012

40. Jafarian, J.H., Takabi, H., Touati, H., Hesamifard, E., Shehab, M.: Towards a general framework for optimal role mining: a constraint satisfaction approach. In: Proceedings of 20th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 211–220, June 2015
41. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering DAC, MAC and RBAC. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 41–55. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31540-4_4
42. John, J.C., Sural, S., Atluri, V., Vaidya, J.S.: Role mining under role-usage cardinality constraint. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) SEC 2012. IAICT, vol. 376, pp. 150–161. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30436-1_13
43. Kern, A., Kuhlmann, M., Schaad, A., Moffett, J.: Observations on the role lifecycle in the context of enterprise security management. In: Proceedings of 7th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 43–51, June 2002
44. Krautsevich, L., Lazouski, A., Martinelli, F., Yautsiukhin, A.: Towards policy engineering for attribute-based access control. In: Bloem, R., Lipp, P. (eds.) INTRUST 2013. LNCS, vol. 8292, pp. 85–102. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03491-1_6
45. Kuhlmann, M., Shohat, D., Schimpf, G.: Role mining - revealing business roles for security administration using data mining technology. In: Proceedings of 8th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 179–186, June 2003
46. Kumar, R., Sural, S., Gupta, A.: Mining RBAC roles under cardinality constraint. In: Proceedings of 6th International Conference on Information Systems Security (ICISS), pp. 171–185, December 2010
47. Krautsevich, L., Lazouski, A., Martinelli, F., Yautsiukhin, A.: Towards attribute-based access control policy engineering using risk. In: Bauer, T., Großmann, J., Seehusen, F., Stølen, K., Wendland, M.-F. (eds.) RISK 2013. LNCS, vol. 8418, pp. 80–90. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07076-6_6
48. Lin, D., Rao, P., Bertino, E., Lobo, J.: An approach to evaluate policy similarity. In: ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 1–10 (2007)
49. Lin, D., Rao, P., Ferrini, P., Bertino, E., Lobo, J.: A similarity measure for comparing XACML policies. IEEE Trans. Knowl. Data Eng. **25**, 1946–1959 (2013)
50. Lu, H., Hong, Y., Yang, Y., Duan, L., Badar, N.: Towards user-oriented RBAC model. In: Proceedings of 27th International Conference on Data and Applications Security and Privacy (DBSec), pp. 81–96, July 2013
51. Lu, H., Hong, Y., Yang, Y., Duan, L., Badar, N.: Towards user-oriented RBAC model. J. Comput. Secur. (JCS) **23**(1), 107–129 (2015)
52. Lu, H., Vaidya, J., Atluri, V.: Optimal Boolean matrix decomposition: application to role engineering. In: Proceedings of 24th IEEE International Conference on Data Engineering (ICDE), pp. 297–306, April 2008
53. Lu, H., Vaidya, J., Atluri, V.: An optimization framework for role mining. J. Comput. Secur. (JCS) **22**(1), 1–31 (2014)
54. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Commun. ACM **19**, 461–471 (1976)
55. Ma, X., Li, R., Lu, Z.: Role mining based on weights. In: Proceedings of 15th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 65–74, June 2010

56. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 335–346. Springer, Heidelberg (2006). https://doi.org/10.1007/11871637_33
57. Mitra, B., Sural, S., Atluri, V., Vaidya, J.: Toward mining of temporal roles. In: Wang, L., Shafiq, B. (eds.) DBSec 2013. LNCS, vol. 7964, pp. 65–80. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39256-6_5
58. Mitra, B., Sural, S., Atluri, V., Vaidya, J.: The generalized temporal role mining problem. J. Comput. Secur. **23**(1), 31–58 (2015)
59. Mitra, B., Sural, S., Vaidya, J., Atluri, V.: Mining temporal roles using many-valued concepts. Comput. Secur. **60**, 79–94 (2016)
60. Mitra, B., Sural, S., Vaidya, J., Atluri, V.: Migrating from RBAC to temporal RBAC. IET Inf. Secur. **11**, 294–300 (2017)
61. Mocanu, D.C., Turkmen, F., Liotta, A.: Towards ABAC policy mining from logs with deep learning. In: International Multiconference (2015)
62. Molloy, I., et al.: Mining roles with semantic meanings. In: Proceedings of 13th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 21–30, June 2008
63. Molloy, I., et al.: Mining roles with multiple objectives. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(4), 36:1–36:35 (2010)
64. Molloy, I., Li, N., Qi, Y.A., Lobo, J., Dickens, L.: Mining roles with noisy data. In: Proceedings of 15th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 45–54, June 2010
65. Molloy, I., Park, Y., Chari, S.: Generative models for access control policies: applications to role mining over logs with attribution. In: Proceedings of 17th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 45–56, June 2012
66. Narouei, M., Khanpour, H., Takabi, H., Parde, N., Nielsen, R.: Towards a top-down policy engineering framework for attribute-based access control. In: ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 103–114 (2017)
67. Narouei, M., Takabi, H.: Towards an automatic top-down role engineering approach using natural language processing techniques. In: Proceedings of 20th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 157–160, June 2015
68. Neumann, G., Strembeck, M.: A scenario-driven role engineering process for functional RBAC roles. In: Proceedings of 7th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 33–42, June 2002
69. O'Connor, A.C., Loomis, R.J.: 2010 economic analysis of Role-Based Access Control. RTI International report for NIST (2010)
70. Roeckle, H., Schimpf, G., Weidinger, R.: Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In: Proceedings of 5th ACM Workshop on Role-Based Access Control (RBAC), pp. 103–110, July 2000
71. Saenko, I., Kotenko, I.: Genetic algorithms for role mining problem. In: Proceedings of 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 646–650, February 2011
72. Saenko, I., Kotenko, I.: Design and performance evaluation of improved genetic algorithm for role mining problem. In: Proceedings of 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 269–274, February 2012

73. Sandhu, R.S.: Lattice-based access control models. Computer **26**(11), 9–19 (1993)
74. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
75. Sarana, P., Roy, A., Sural, S., Vaidya, J., Atluri, V.: Role mining in the presence of separation of duty constraints. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2015. LNCS, vol. 9478, pp. 98–117. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26961-0_7
76. Shin, D., Ahn, G., Cho, S., Jin, S.: On modeling system-centric information for role engineering. In: Proceedings of 8th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 169–178, June 2003
77. Smolensky, P.: Information processing in dynamical systems: foundations of harmony theory. In: Parallel Distributed Processing, pp. 194–281 (1987)
78. Strembeck, M.: Scenario-driven role engineering. IEEE Secur. Priv. **8**(1), 28–35 (2010)
79. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: Proceedings of 12th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 175–184, June 2007
80. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: a formal perspective. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(3), 27:1–27:31 (2010)
81. Vaidya, J., Atluri, V., Guo, Q., Lu, H.: Edge-RMP: minimizing administrative assignments for role-based access control. J. Comput. Secur. (JCS) **17**(2), 211–235 (2009)
82. Vaidya, J., Atluri, V., Warner, J.: Role miner: mining roles using subset enumeration. In: Proceedings of 13th ACM Conference on Computer and Communications Security (CCS), pp. 144–153, October 2006
83. Vaidya, J., Atluri, V., Warner, J., Guo, Q.: Role engineering via prioritized subset enumeration. IEEE Trans. Dependable Secur. Comput. (TDSC) **7**(3), 300–314 (2010)
84. Vaidya, J., Shafiq, B., Atluri, V., Lorenzi, D.: A framework for policy similarity evaluation and migration based on change detection. Network and System Security. LNCS, vol. 9408, pp. 191–205. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25645-0_13
85. Verde, N.V., Vaidya, J., Atluri, V., Colantonio, A.: Role engineering: from theory to practice. In: Proceedings of 2nd ACM Conference on Data and Application Security and Privacy (CODASPY), pp. 181–191, February 2012
86. Xu, Z., Stoller, S.D.: Algorithms for mining meaningful roles. In: Proceedings of 17th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 57–66, June 2012
87. Xu, Z., Stoller, S.: Mining attribute-based access control policies from logs. Computing Research Repository - arXiv (2014)
88. Xu, Z., Stoller, S.: Mining attribute-based access control policies. IEEE Trans. Dependable Secur. Comput. (TDSC) **12**, 533–545 (2015)
89. Zhang, D., Ramamohanarao, K., Ebringer, T.: Role engineering using graph optimisation. In: Proceedings of 14th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 139–144, June 2007
90. Zhang, D., Ramamohanarao, K., Ebringer, T.: Permission set mining: discovering practical and useful roles. In: Proceedings of 24th Annual Computer Security Applications Conference (ACSAC), pp. 247–256, December 2008
91. Zhang, W., Chen, Y., Gunter, C., Liebovitz, D., Malin, B.: Evolving role definitions through permission invocation patterns. In: Proceedings of 18th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 37–48, June 2013