Anomaly Detection and Visualization in Generative RBAC Models

Maria Leitner[†*]
maria.leitner@univie.ac.at

Stefanie Rinderle-Ma[†]
stefanie.rinderle-ma@univie.ac.at

[†]University of Vienna, Faculty of Computer Science, Vienna, Austria
[*]SBA Research, Vienna, Austria

ABSTRACT

With the wide use of Role-based Access Control (RBAC), the need for monitoring, evaluation, and verification of RBAC implementations (e.g., to evaluate ex post which users acting in which roles were authorized to execute permissions) is evident. In this paper, we aim at detecting and identifying anomalies that originate from insiders such as the infringement of rights or irregular activities. To do that, we compare prescriptive (original) RBAC models (i.e. how the RBAC model is expected to work) with generative (current-state) RBAC models (i.e. the actual accesses represented by an RBAC model obtained with mining techniques). For this we present different similarity measures for RBAC models and their entities. We also provide techniques for visualizing anomalies within RBAC models based on difference graphs. This can be used for the alignment of RBAC models such as for policy updates or reconciliation. The effectiveness of the approach is evaluated based on a prototypical implementation and an experiment.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.4 [**Management of Computing and Information Systems**]: System Management—*Management audit*

General Terms

Measurement, Security

Keywords

access control; anomaly detection; audit; graph edit distance; inexact graph matching; RBAC; similarity

1. INTRODUCTION

Enterprise management systems use Role-based Access Control (RBAC) as de facto standard to enforce access control policies (cf. [11]). Currently, there are two general approaches to establish RBAC models. In the *top-down* approach, the RBAC models are typically derived by analyzing e.g., business processes and artifacts in order to obtain roles that can fulfill the business operations adequately (cf. [22]). In the *bottom-up* approach, role mining techniques are used to automatically generate RBAC models based on an access control configuration (e.g., a set of user-permissions) such as in [14, 24, 18]. These generative RBAC models contain roles that may indicate job functions or organizational entities.

Recent developments show that generative RBAC models can be derived from event logs (e.g., [2, 19]). For example, Molloy et al. [19] apply role mining techniques on event logs to generate RBAC models. Another approach in Baumgrass et al. [2] proposes to derive role engineering artifacts from event logs. These generative (current-state) RBAC models contain which users in which roles have actually invoked which permissions i.e., they reflect operational reality. Hence, these access snapshots can be used for an ex post analysis and evaluation of RBAC implementations.

In this paper, we aim at detecting potential anomalies in RBAC models that originate from insiders. Anomalies represent patterns in event logs that do not conform to the specified notion of normal behavior (cmp. [6]). In our case, the normal behavior is specified in an original (prescriptive) RBAC model (i.e. it specifies how users are expected to work) and we compare it with a generative model (i.e. how users actually work) obtained with e.g., role mining techniques. The generative model might deviate from the prescriptive model that has been deployed in the enterprise management system. These deviations may occur due to e.g., policy errors, misconfiguration or changed user behavior (which might indicate insider threats).

Anomalies can be distinguished into *point*, *contextual*, and *collective* anomalies (see [6]). In this paper, we center on the detection of point and contextual anomalies. Point anomalies occur due to structural modifications between prescriptive and generative RBAC models. Examples are new role-permission or user-role assignments. Such point anomalies can be detected based on comparing the *structures* of prescriptive and generative RBAC models. We provide measures based on set-based and graph-based representations of RBAC models.

Contextual anomalies go beyond structural differences between RBAC models, i.e., it is possible that two RBAC models differ structurally, but not in a contextual manner. Assume, for example, that in the prescriptive RBAC model a user u has role r which is assigned two permissions p_1 and p_2. Assume further that in the generative RBAC model,

SACMAT'14, June 25–27, 2014, London, ON, Canada

http://dx.doi.org/10.1145/2613087.2613105.

role r is split into two roles r_1 with permission p_1 and role r_2 with permission p_2 and u is assigned both roles r_1 and r_2. Then prescriptive and generative RBAC model obviously differ structurally, but from a point of view of user u she still has the same set of permissions as before. Analyzing contextual anomalies on top of structural differences can yield insights on the actual effects of the deviations of the generative RBAC model from the prescriptive one.

In order to analyze and interpret the differences between RBAC models with respect to anomalies it is crucial to convey the information about the differences to the analyst. For this we elaborate on techniques for visualizing the difference between RBAC models. We employ difference models [1] as means to build RBAC difference as well as similarity models.

Our approach is outlined as follows: In Section 2, we define the fundamentals of RBAC models and their transformation onto directed acyclic graphs. We provide different similarity metrics for measuring structural differences in Section 3 and contextual differences between RBAC models in Section 4 along with their algorithmic realization in Section 5. Section 6 provides implementation details, visualization techniques, and experimental results. Related work is outlined in Section 7. Section 8 concludes the paper.

2. PRELIMINARIES

We base our approach on the NIST standard RBAC model (cf. Definition 1) [11] and its administrative operations such as `addUser` or `assignUser` (further called edit operations) as defined in [11]. The edit operations provide one option to describe differences between two RBAC models. The other option is to capture the differences based on the graph representation in a set-based manner.

Definition 1 (RBAC Model). *Based on Ferraiolo et al. [11], an RBAC model RBAC is defined as* $RBAC := (U, R, P, UA, PA, RH)$ *with: U is a set of users, R is a set of roles, P is a set of permissions, $UA \subseteq U \times R$ is a many-to-many relation between users and roles, $PA \subseteq R \times P$ is a many-to-many mapping between roles and permissions, and $RH \subseteq R \times R$ is an inheritance relation where $r_1 \succeq r_2$ only if all permissions of r_2 are also permissions of r_1 and all users assigned to r_1 are also users authorized to r_2.*

Based on RBAC models, a set of review functions [11] is useful for defining similarity metrics later. We summarize these functions in the appendix in Definition 7.

For comparison and particularly visualization reasons, we transform RBAC models into graphs (cmp. [12]). In particular, we will transform an RBAC model into a directed acyclic graph (DAG) (see [9]) in order to specify role hierarchy relations (i.e., the direction from senior roles to junior roles).

Definition 2 (Graph-transformed RBAC Model). *Let* $RBAC := (U, R, P, UA, PA, RH)$ *be an RBAC model. Then the graph-transformed RBAC model is defined as* $RM := (N, E)$ *with:*

- $N := \{U \cup R \cup P\}$ *is a set of nodes*
- $E := \{UA \cup PA \cup RH\}$ *is a set of edges*

In our approach, we assume that the graph-transformed RBAC models utilize **unique** labels. This is justified by the application as access control systems use unique IDs for users, roles, and permissions. In addition, the labels of edges are represented by the labels of the (two) nodes the edges (n, m) interconnect.

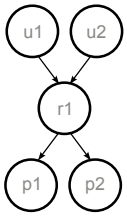

Figure 1: Example: Graph-transformed Representation of an RBAC model

For example, Figure 1 shows a graph-transformed representation of an RBAC model with two users u_1 and u_2, one role r_1, and two permissions p_1 and p_2. The users are assigned role r_1 by directed edges (u_1, r_1) and (u_2, r_1) and role r_1 is assigned two permissions by edges (r_1, p_1) and (r_1, p_2).

Furthermore, we claim that graph RM does not contain isolated nodes. In this paper, we assume that the basis for generative RBAC models are event logs which consist typically of user-permission tuples (cf. [2]). Hence, in our approach, each node must have a relation with another node, i.e. a user or permission node cannot exist without a role assignment and a role node cannot exist without the assignment of a user and permission.

3. STRUCTURAL ANALYSIS

In this paper, we focus on identifying insider threats from advanced users that are typically trusted individuals who know the internal organization and application systems. Their application use is recorded in an event log. Based on the data, we derive a generative RBAC model and analyze patterns that differ from the normal behavior (as defined in the prescriptive RBAC model).

In this section, we tackle the problem in two ways: in Section 3.1, we analyze potential anomalies between graph-transformed representations of two RBAC models in a set-based manner. In Section 3.2, we investigate the application of graph matching techniques for the graph-transformed representations of RBAC models for analysis of similarities and differences between the underlying RBAC models.

3.1 Potential Point Anomalies

As can be seen in Figure 2, we target at three types of modifications in RBAC models: missing vertices, additional nodes, and connectivity change in graph-transformed RBAC models. We also show how these anomalies can be detected based on the graph-transformed representation of RBAC models.

Missing Vertices.

As generative RBAC models reflect operational reality, they only consist of actually "used" roles, permissions, and users. For example, if employees are on vacation, their users might not show up in the generative RBAC model. Furthermore, permissions may not be included or are "missed" because they have not been in use. Given a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed

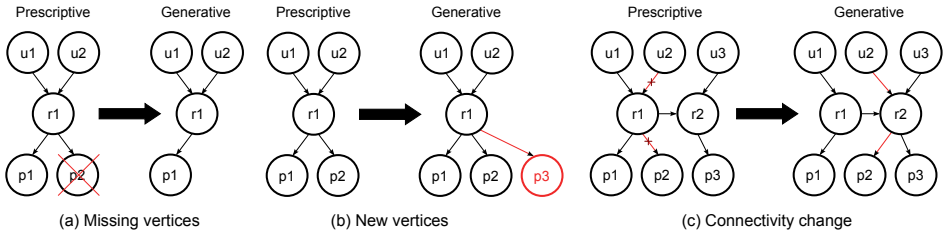

Figure 2: Potential Anomalies in Generative RBAC Models

representation $RM_2 = (N_2, E_2)$, the set of missing nodes or vertices N_{mis} can be determined by $N_{mis} := N_1 \setminus N_2$.

For example, the president has the entitlement to order the use of nuclear weapons (permission p_2) as shown in Figure 2(a). Albeit he might not make use of the entitlement, he/she still has the "permission". However, the permission p_2 is not shown in the generative RBAC model as it has not been utilized (cf. Figure 2(a)). Hence, as generative models might not contain a full set of access control policies (as defined in the prescriptive models), a missing node might not automatically be evaluated as anomaly but more as a deficiency/ absence of permissions. In other cases, the absence of permissions might indicate abnormal (intended) behavior for e.g., permissions that are executed in daily business processes.

New Vertices.

Anomalies that stem from the infringement of rights may result into new vertices in the RBAC graph. Given a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed representation $RM_2 = (N_2, E_2)$, the set of new nodes or vertices N_{new} can be determined by $N_{new} := N_2 \setminus N_1$.

For example, new vertices (that cannot be matched to the original model) may stand for new employees (users) or new functional roles. However, in some cases they may represent the use of "fake" user profiles (users) or roles. Furthermore, new permissions (assigned to roles) may indicate that roles are granted more privileges (e.g., the use of new applications) or that the role has been empowered illicitly. As shown in Figure 2(b), role *r1* is empowered by granting an additional privilege p_3. Hence, users u_1 and u_2 are authorized to perform p_3.

Connectivity Change.

Connectivity change in an RBAC graph can result from e.g., rights infringement or changing user behavior (due to e.g., promotions, demotions). Given a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed representation $RM_2 = (N_2, E_2)$, connectivity changes can be determined by the set of missing or new edges in RM_2, i.e., $E_{cch} := (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$.

In Figure 2(c), the role membership of user u_2 and the permission assignment of p_2 changes. Reasons for this connectivity change can be such as promotions (e.g., user u_2 has been promoted to role r_2), new projects (e.g., user u_2 is now part of project r_2), or security violations (e.g., user u_2 has illicitly acquired role r_2). A connectivity change may look simple but can have a large effect on the RBAC model. It may be harder to interpret than e.g., missing or new vertices.

Analyzing the missing and new nodes and connectivity change in the RBAC models can indicate potential anomalies. However, to identify potential anomalies as actual insider threats is a challenging task. To do that, the final evaluation of anomalies and their interpretation such as insider threats, violations, or organizational changes can only be performed with a certain level of domain knowledge. However, with our approach we want to support the selection of potential anomalies for further investigation.

3.2 Graph-matching Distance Measures

Based on the graph-transformed RBAC representation (cf. Definition 2), the structure of the RBAC model can be also investigated using graph matching techniques. We evaluated *exact* and *inexact* graph matching techniques (see [8]). While *exact* matching techniques require a strict correspondence between two graphs being matched, *inexact* graph matching techniques do not require two graphs to be identical i.e., the comparison of completely different graphs is possible. As we aim to compare not completely identical graph-transformed RBAC models, *inexact* graph matching techniques are highly suitable for the structural analysis of RBAC models. In fact, we could apply the problems graph isomorphism, subgraph isomorphism, and the maximum common subgraph (cf. [5, 9]) for the purpose of comparing two RBAC models.

Graph isomorphism determines whether there exists a bijective mapping between the node and edges sets of the graph-transformed representations of two RBAC models, i.e., it can be used to check whether two RBAC models are "the same". As we assume a unique node labeling for RBAC models and their graph-transformed representations, we can check equality of node and edge sets, i.e., for a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed representation $RM_2 = (N_2, E_2)$, $RM_1 = RM_2$ holds if $N_1 = N_2$ and $E_1 = E_2$.
Interpretation: If $RM_1 = RM_2$ holds, no deviations from the prescriptive model have occurred, i.e., no anomalies within the generative models are detected.

Subgraph isomorphism can be utilized to identify if a graph-transformed representation of an RBAC model is a part of a graph-transformed representation of another RBAC model. Assume that a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed representation $RM_2 = (N_2, E_2)$ are given. We can determine that $RM_1 \subset RM_2$ if $N_1 \subset N_2$ and $E_1 \subset E_2$ (and vice versa) assuming unique node labeling.
Interpretation: If $RM_1 \subset RM_2$, new vertices and connectivities have been added to RM_1. This corresponds to anomaly (b) – new vertices – as depicted in Figure 2. If $RM_2 \subset RM_1$,

parts of RM_1 have been "removed", i.e., vertices are missing corresponding to anomaly (a) as shown in Figure 2.

The maximum common subgraph can be used to compare the similarity of two graphs even if they are not isomorph or subgraph isomorph. For a prescriptive RBAC model with graph-transformed representation $RM_1 = (N_1, E_1)$ and a generative RBAC model with graph-transformed representation $RM_2 = (N_2, E_2)$, the maximum common subgraph $mcs(RM_1, RM_2)$ is a subgraph of RM_1 and RM_2 and there exists no subgraph of RM_1 and RM_2 that is larger than $mcs(RM_1, RM_2)$. For a formal definition we refer to [4]. Interpretation: The larger the maximum common subgraph is, the higher is the similarity between two graphs. This can be seen as a general indicator of how similar two RBAC models are or, in turn, how much they differ from each other.

In the following, we will analyze distance measures for the maximum common subgraph problem in order to compare different graph-transformed representations of RBAC models. The first measure is the *graph edit distance* $d_{ged}(RM_1, RM_2)$ for graph-transformed RBAC models RM_1 and RM_2 that is the minimal number of graph edit operations necessary to transform RM_1 into RM_2 [4]. Edit operations may include node/ edge insertions and deletions. In our approach, we consider the RBAC edit operations outlined in Section 2 as graph edit operations. For this, [4] suggest the following formula that transforms the calculation of the graph edit distance to computing the maximum common subgraph:

$$d_{ged}(RM_1, RM_2) = |RM_1| + |RM_2| - 2\,|mcs(RM_1, RM_2)| \tag{1}$$

Interpretation: Based on edit operations, it can be precisely stated which modifications have been applied to transform RM_1 to RM_2. Hence, the shorter the sequence is to transform one graph to another, the greater is their similarity. If RM_1 was transformed into RM_2 intentionally by, for example, executing some optimizations on RM_1 and if the edit operations were logged, they can easily be analyzed. However, if we derive generative RBAC models from logs, we do not know the edit operations that might have transformed the prescriptive model into the generative one. Then, $d_{ged}(RM_1, RM_2)$ has to be determined on basis of RM_1 and RM_2.

Furthermore, determining the graph edit operations indicates how many operations it would take to transform the prescriptive into the generative RBAC model but does not express the similarity of the two models. In relation to the previous measure, another distance metric d_{mcs} based on the maximum common subgraph is defined in [5] as follows:

$$d_{mcs}(RM_1, RM_2) = 1 - \frac{|mcs(RM_1, RM_2)|}{max(|RM_1|, |RM_2|)} \tag{2}$$

where mcs is the maximum common subgraph, $|\ldots|$ is the size of a graph[1], and $max(\ldots)$ is the maximum operation. Another distance measure is based on the idea of graph union in [26]:

$$d_{gu}(RM_1, RM_2) = 1 - \frac{|mcs(RM_1, RM_2)|}{|RM_1| + |RM_2| - |mcs(RM_1, RM_2)|} \tag{3}$$

Interpretation: The distance metrics d_{mcs} and d_{gu} provide a value, i.e., how much the generative RBAC model deviates from the prescriptive one, but not the exact graph edit operations. Having both measures is an asset for the anomaly analysis.

In later sections, we will show how these measures are used to identify point anomalies.

[1]The size of a graph-transformed representation RM can be determined by $|RM| := |N| + |E|$.

4. SEMANTIC ANALYSIS

When we first investigated the similarity of RBAC models, we discovered that not only the structure can indicate anomalies and security violations, but also the semantic aspects of RBAC models are important such as the semantic meaning of roles (cf. [14, 18]) as they are typically designed for a purpose such as to fulfill a job function. For instance, Figure 3 displays two RBAC models which look very similar at first glance. In *RBAC1*, users u_1 and u_2 are assigned to roles r_1 and r_2. However, in *RBAC2* user u_1 is only assigned to role r_1 and user u_2 only to r_2. It can be seen by evaluating the review functions (see Definition 7 in the appendix) that both models differ structurally and have a different role composition. However, an evaluation of the access control configurations shows that they are identical (i.e. both models have the same set of user-permissions). Even tough both models differ structurally, they enable the same "behavior" for users because of the same user-permissions. Hence, these contextual aspects should be considered when comparing RBAC models semantically.

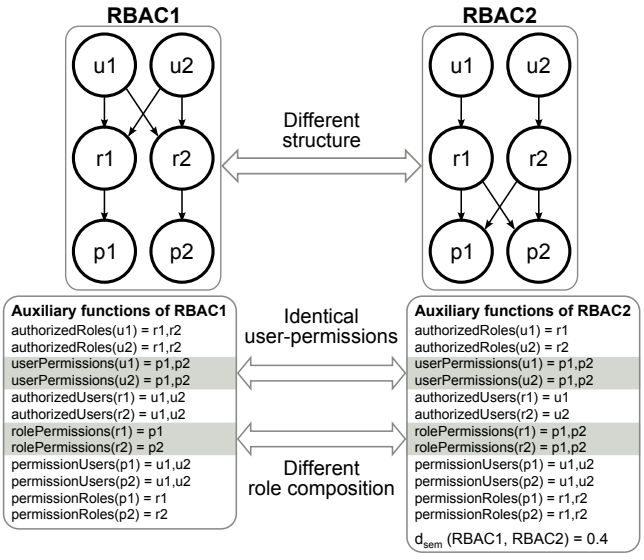

Figure 3: Example for Semantic Differences

In our analysis, we compare "the composition" of RBAC models; we investigate the associations of users, roles and permissions in the RBAC models. For example, for each user node in the generative model we analyze if the user has the same authorized roles and permissions as in the original model (cf. Figure 3). If these authorized roles and permissions deviate from the original model, these semantic differences can be identified as potential *contextual* anomalies. We were inspired by [25, 23] who measure the similarity between role sets with the Jaccard coefficient. We further adapt it to analyze not only the similarity of roles but also of users and permissions which are further used to compute the semantic similarity of RBAC models.

Approach

In the following, we will assess the semantic similarity for RBAC models: a predictive RBAC model $RBAC_1 = (U_1, R_1, P_1, UA_1, PA_1, RH_1)$ and a current-state RBAC model $RBAC_2 = (U_2, R_2, P_2, UA_2, PA_2, RH_2)$. Thereby, we will compare model $RBAC_2$ with model $RBAC_1$ in order to determine semantic anomalies. We will use the Jaccard Coefficient which is a well known similarity metric (cf. [24, 23]) to compare the similarity between node sets. For the computation of a semantic similarity measure between RBAC models, we calculate the similarity of each node in the model and use them to calculate the semantic similarity. The overall approach is outlined as follows:

1. In the first step, the nodes of models $RBAC_2$ and $RBAC_1$ are matched by their (unique) labels.
2. For each matched node, the node similarity is calculated. For each "unmatched" node, the similarity is set to a threshold value. In this approach, we set the similarity value for unmatched nodes to zero.
3. Then, we calculate the similarity measure for each user, role, and permission node.
4. Finally, we compute the average semantic similarity measure based on the node similarity measures.

In the next paragraphs, we will describe the computation of the semantic similarity measure between $RBAC_1$ and $RBAC_2$ in detail. Recall Definition 7 in the appendix for review functions on RBAC models.

User Node Analysis.

In the first step, the similarity of all user nodes are analyzed based on their authorized roles and user-permission assignments. For each pair of matched user nodes (u_i, u_j) (i.e., $u_i = u_j, i \neq j$), we determine the user-role similarity $simUR$ and the user-permission similarity $simUP$. The user-role similarity measure $simUR$ between users u_i and u_j is defined by the authorized roles for users:

$$simUR(u_i, u_j) = \frac{|\texttt{authorizedRoles}(u_i) \cap \texttt{authorizedRoles}(u_j)|}{|\texttt{authorizedRoles}(u_i) \cup \texttt{authorizedRoles}(u_j)|}$$

The function $\texttt{authorizedRoles}(u_i)$ returns a set of authorized roles for user $u_i \in U_1$ of model RM_1 and $\texttt{authorizedRoles}(u_j)$ returns a set of authorized roles for user $u_j \in U_2$ of model RM_2 (cf. Definition 7).

The user-permission similarity $simUP$ between users u_i and u_j (if $u_i = u_j, i \neq j$) is defined based on the user permissions where $\texttt{userPermissions}(u_i)$ is the set of all permissions authorized for user $u_i \in U_1$ and $\texttt{userPermissions}(u_j)$ is the set of all permissions authorized for user $u_j \in U_2$ (cf. Definition 7):

$$simUP(u_i, u_j) = \frac{|\texttt{userPermissions}(u_i) \cap \texttt{userPermissions}(u_j)|}{|\texttt{userPermissions}(u_i) \cup \texttt{userPermissions}(u_j)|}$$

Furthermore, we will combine these measures to specify the similarity measure between two user nodes.

Definition 3 (User-User Similarity). *The user-user similarity userSim for two users u_i and u_j with $u_i = u_j, i \neq j$ is the weighted sum of the user-role and the user-permission similarity:*

$$userSim(u_i, u_j) = w_{ur} * simUR + w_{up} * simUP$$

where $w_{ur}, w_{up} \in [0; 1] \wedge w_{ur} + w_{up} = 1$

In our paper, we specified the weights as $w_{ur} = w_{up} = \frac{1}{2}$. The function $userSim$ returns a value between 0 and 1. If two user nodes are identical (i.e. have the same associations with roles and permissions) the similarity is 1. The more identical both user nodes are (i.e. have the same set of authorized roles and user-permissions) the higher is their similarity. Furthermore, we can compute the distance between user nodes as $d_{user}(u_i, u_j) = 1 - userSim(u_i, u_j)$.

Role Node Analysis.

In a next step, we investigate all role nodes. To do that, we adapted the role-permission, role-user, role hierarchy relation, and role-role similarity measure defined in [23] to our approach. In the following, we will describe the role-user similarity $simRU$, role hierarchy relation $simRH$ and role-permission $simRP$ similarity and use them as a composite measure for the similarity between roles.

The role-user similarity between two roles r_i and r_j (for a match $r_i = r_j, i \neq j$) is specified based on the authorized users for roles [23]; the function $\texttt{authorizedUsers}(r_i)$ returns a set of authorized users for role $r_i \in R_1$ and $\texttt{authorizedUsers}(r_j)$ returns a set of authorized users for role $r_j \in R_2$ (cf. Definition 7).

$$simRU(r_i, r_j) = \frac{|\texttt{authorizedUsers}(r_i) \cap \texttt{authorizedUsers}(r_j)|}{|\texttt{authorizedUsers}(r_i) \cup \texttt{authorizedUsers}(r_j)|}$$

The role hierarchy relation similarity $simRH$ is defined based on the hierarchy relations (senior and junior roles) for roles [23]. The function $\texttt{senior}(r_i)$ returns a set of senior roles of role r_i and $\texttt{junior}(r_i)$ returns a set of junior roles of role r_i. While in [23] only immediate senior and junior roles are returned, in our approach full inheritance (comp. [18]) is applied i.e., all senior and junior roles associated to a role are returned (cf. Definition 7).

$$simRH(r_i, r_j) = w_{sen} * \frac{min(|\texttt{senior}(r_i)|, |\texttt{senior}(r_j)|)}{max(|\texttt{senior}(r_i)|, |\texttt{senior}(r_j)|)} + w_{jun} * \frac{min(|\texttt{junior}(r_i)|, |\texttt{junior}(r_j)|)}{max(|\texttt{junior}(r_i)|, |\texttt{junior}(r_j)|)}$$
$$\text{where } w_{sen}, w_{jun} \in [0; 1], w_{sen} + w_{jun} = 1$$

The role-permission similarity $simRP$ is specified based on the set of permissions each role has [23]. The function $\texttt{rolePermissions}(r_i)$ returns all permissions role $r_i \in R_1$ has and $\texttt{rolePermissions}(r_j)$ returns all permissions associated to role $r_j \in R_2$ (cf. Definition 7).

$$simRP(r_i, r_j) = \frac{|\texttt{rolePermissions}(r_i) \cap \texttt{rolePermissions}(r_j)|}{|\texttt{rolePermissions}(r_i) \cup \texttt{rolePermissions}(r_j)|}$$

Based on these measures, we determine the similarity measure between two matching roles, i.e., $r_i = r_j, i \neq j$.

Definition 4 (Role-Role Similarity). *We combine the role-user, role hierarchy relation and role-permission similarity to specify the role-role similarity roleSim (cf. [23]).*

$$roleSim(r_i, r_j) = w_{ru} * simRU + w_{rh} * simRH + w_{rp} * simRP$$
$$where\ w_{ru}, w_{rh}, w_{rp} \in [0; 1] \wedge w_{ru} + w_{rh} + w_{rp} = 1$$

In our approach, we set the weights $w_{ru} = w_{rh} = w_{rp} = \frac{1}{3}$. The function $roleSim \in [0, 1]$ returns a value between 0 and 1. If two role nodes are identical their similarity is 1. Role nodes with only some common features have a value between 0 and 1. Role nodes have a high similarity if they have a large set of identical authorized users, role-permissions, and hierarchy relations. Additionally, we can define the distance between role nodes as $d_{role}(r_i, r_j) = 1 - roleSim(r_i, r_j)$.

Permission Node Analysis.

In the next step, the similarity of matched permission nodes (p_i, p_j) with $p_i = p_j, i \neq j$ is computed based on users and roles that are authorized to access the permission. The permission-user similarity $simPU$ is specified based on the authorized users to a permission; $\texttt{permissionUsers}(p_i)$ returns a set of authorized users for permission $p_i \in P_1$ and $\texttt{permissionUsers}(p_j)$ returns a set of authorized users for permission $p_j \in P_2$.

$$simPU(p_i, p_j) = \frac{|\texttt{permissionUsers}(p_i) \cap \texttt{permissionUsers}(p_j)|}{|\texttt{permissionUsers}(p_i) \cup \texttt{permissionUsers}(p_j)|}$$

Furthermore, we compute the permission-role similarity $simPR$ based on the authorized roles to a permission. The function $\texttt{permissionRoles}(p_i)$ returns a set of authorized roles for permission $p_i \in P_1$ and $\texttt{permissionRoles}(p_j)$ returns a set of authorized roles for permission $p_j \in P_2$.

$$simPR(p_i, p_j) = \frac{|\texttt{permissionRoles}(p_i) \cap \texttt{permissionRoles}(p_j)|}{|\texttt{permissionRoles}(p_i) \cup \texttt{permissionRoles}(p_j)|}$$

Based on these measures, we compute the similarity between two permissions.

Definition 5 (Permission-Permission Similarity). *The similarity for any two permissions $p_i \in P_1$ and $p_j \in P_2$ with $p_i = p_j, i \neq j$ is specified as the weighted sum of the permission-users $simPU$ and the permission-roles $simPR$ similarity.*

$$permSim(p_i, p_j) = w_{pu} * simPU + w_{pr} * simPR \text{ where } w_{pu} + w_{pr} = 1$$

The function $permSim$ returns a value between 0 and 1. The similarity of corresponding permission nodes increases the more user-permission and permission-roles are matched. Subsequently, we can specify the distance between permission nodes as $d_{perm}(p_i, p_j) = 1 - permSim(p_i, p_j)$.

Based on these similarity measures of the users, roles, and permissions, we can further specify the semantic similarity.

Definition 6 (Semantic Similarity).

$$semSim(RBAC_1, RBAC_2) = \frac{1}{m} \sum_{i=0}^{m} sim(n_i)$$
$$\text{where } n_i \in U_1 \cup U_2 \cup R_1 \cup R_2 \cup P_1 \cup P_2 \quad (4)$$

See Definition 8 in the appendix for a specification of $sim(n_i)$.

Accordingly, the more the user, role, and permission nodes of one model correspond to the nodes in the other model, the higher is their semantic similarity. The similarity function $semSim(RBAC_1, RBAC_2) \in [0, 1]$ has value 1 if $RBAC_1$ and $RBAC_2$ are identical and value 0 if $RBAC_1$ and $RBAC_2$ share no common features. Furthermore, we can define the semantic distance between two RBAC models as $d_{sem}(RBAC_1, RBAC_2) = 1 - semSim(RBAC_1, RBAC_2)$.

With determining the semantic similarity between two RBAC models, we aim to detect anomalies in the composition of RBAC models. Missing or new vertices and connectivity changes (cf. Section 3.1) affect each node similarity and therefore also the semantic similarity ($semSim$). For example, the user-user similarity ($simU$) incorporates if a user node is newly connected to a role or "misses" a role (by the user-role similarity $simUR$).

In the next sections, we will show how the semantic analysis is utilized to detect and identify potential anomalies.

5. ALGORITHMS

This section outlines the algorithms proposed for the structural and semantic metrics that were described in previous sections.

The algorithm to compute the distance measures for the structural analysis is straightforward. Because of the assumption that graph-transformed RBAC models use unique labels, the implementation of the distance measures requires only the intersection of sets and a function that returns the cardinality of a set. Hence, to compute the distance measures, all vertices and edges have to be analyzed in both RBAC models which takes $O(|E_1| + |E_2|)$ time.

Furthermore, the algorithm to compute the semantic distance between two RBAC models is displayed in the appendix in Algorithm 1 and basically summarizes the semantic analysis from the previous section. In the first step, the labels of all nodes are matched between RBAC models RM_1 and RM_2. For each node that has been matched, the node similarity is computed. In particular, the user-user similarity for user nodes, the role-role similarity for role nodes, and the permission-permission similarity for permissions. For each unmatched node, the similarity is set to a threshold value. Then, the average of all similarity values is calculated and the semantic distance is computed by subtracting one from the average.

The review functions (cf. Definition 7 in the appendix) to analyze the relations of the nodes such as `authorizedRoles`, `authorizedUsers`, and `userPermissions` return the cardinality of a set of nodes. We implemented each function as a depth-first search (DFS). For example, to determine the user-user similarity for user u_1 in the predictive RBAC model in Figure 2(c), we have to analyze the authorized roles and the user-permissions with a DFS. The DFS analyzes the associated (outgoing) edges of user node u_1 which leads to role r_1. In the next steps, the outgoing edges of r_1 are analyzed and p_1,p_2, and r_2 are discovered. Subsequently, permission p_3 (associated with r_2) is discovered. Hence, for user u_1 the authorized roles are r_1 and r_2 and the user-permissions are p_1, p_2, and p_3. These authorized roles and user-permissions of the predictive model are compared to the authorized users and user-permissions from the generative model (`authorizedRoles`: r_1, r_2, `userPermissions`: p_1, p_2, p_3) and the user-user similarity for node u_1 is computed ($userSim(P.u_1, G.u_1) = 1$). Unfortunately, the use of DFS and the matching of nodes increases the time to compute the semantic distance measure to $O(N^3)$ with $N := N_1 + N_2$. However, this opens the possibility for further optimizations such as with approximate solutions in future research.

6. EVALUATION

This section describes the implementation, visualization and experimental results of this approach.

6.1 Implementation

We used graph-tools (cf. `http://graph-tool.skewed.de/`), an efficient Python module for manipulation and statistical analysis of graphs to implement our approach. In particular, we have implemented the proposed structural and semantic analysis of RBAC models. Furthermore, the prototype represents RBAC models as graphs as shown in Example 1.

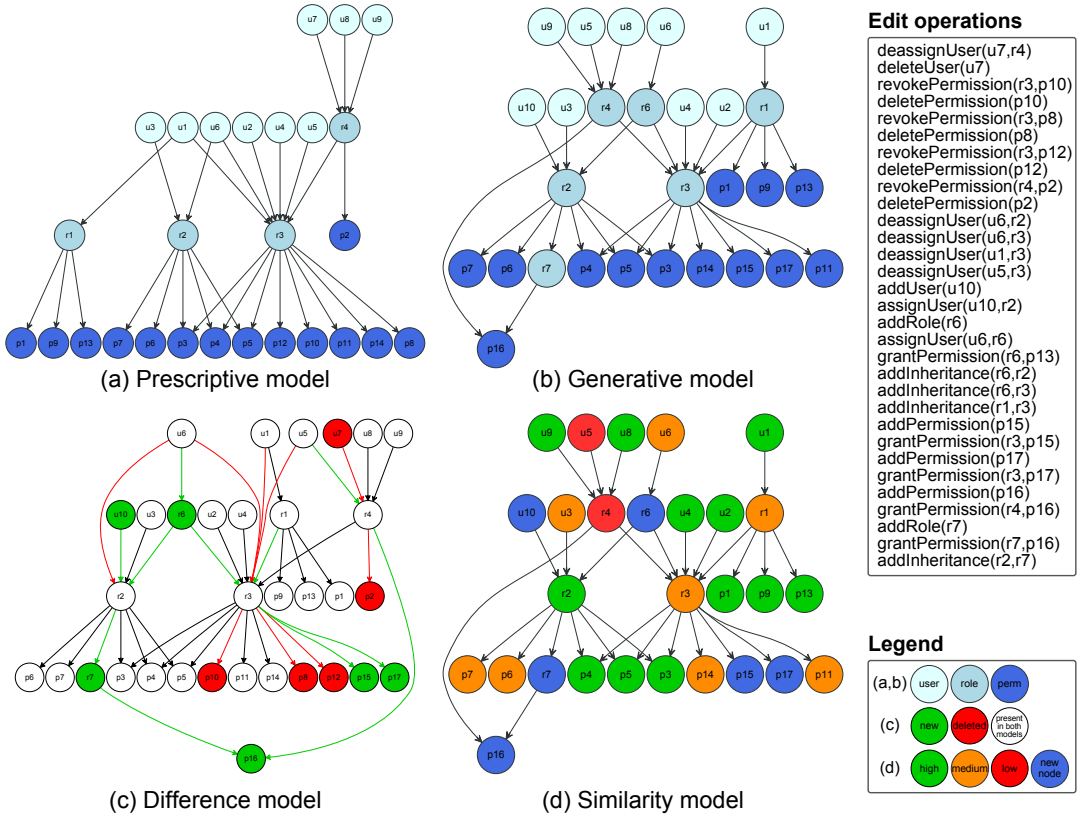

Figure 4: Example 1 consists of (a) a Predictive RBAC Model and (b) a Generative RBAC model and their (c) Difference Model and (d) Similarity Model

Example 1. The example in Figure 4 displays a prescriptive RBAC model in Figure 4(a) and a generative RBAC model in (b). As can be seen from the figures, both models contain user, role, and permission nodes. Edges represent user-role assignments (UA), role hierarchy relations (RH), and role-permission assignments (PA). In addition, we use shades of blue to distinguish between the type of node: user, role, and permission (only for the purpose of representation).

For illustration reasons, we selected an RBAC model of small size as example. However, as RBAC models can become very complex, the visualization of RBAC models is a challenging task. We are aware that the visualization of large-scale RBAC models might result into very large and complex images. In this case, a section of the RBAC model (e.g., visualize only a set of roles and their assignments) can reduce the complexity for illustration purposes or other visualizations might be more applicable (e.g., [28, 3]).

When comparing the RBAC models in Figure 4(a) and (b), it is noticeable that there are differences between them. However, it is hard to clearly detect the structural differences just by looking at both figures. Furthermore, it is even tougher to identify if there are semantically similar or not. That is why we developed visualizations to display structural and semantic differences between RBAC models.

6.2 Visualization of Potential Anomalies

With the use of a visualization, we aim to support the easier identification and understanding of potential anomalies in order to evaluate insider threats (see [16]). Furthermore, the visualization should facilitate the investigation of RBAC differences such as policy updates or reconciliation. In this paper, we propose two models to display anomalies in RBAC models of moderate size: a difference model that highlights *point* anomalies and a similarity model that reveals *contextual* anomalies.

Difference Model.

Typically, a difference model is utilized to contrast or compare two distinct models (e.g., [1, 13]). In our paper, we use it to display the graph edit operations necessary to transform one model into the other (i.e. to visualize structural differences). Let $RM_1 = (N_1, E_1)$ and $RM_2 = (N_2, E_2)$ be two graph-transformed RBAC models (cf. Section 2). The difference model DM can be specified as:

$$DM := (N_1 \cup N_2, E_1 \cup E_2, M_N, M_E) \tag{5}$$

The markings of nodes M_N and edges M_E of the difference model describe if a node/an edge is changed, i.e., if a node/an edge is added or deleted. The node markings distinguish between (0) a node with the same label exists in both models, (1) a node that exists only in the generative model RM_2 (i.e. it has been added), or (-1) a node that exists only in the predictive model RM_1 (i.e. it has been deleted). For each node $n \in N_1 \cup N_2$ the marking is defined as follows:

$$M_N(n) = \begin{cases} 0 & \text{if } n \in N_1 \wedge n \in N_2 \\ 1 & \text{if } n \in N_2 \wedge n \notin N_1 \\ -1 & \text{if } n \in N_1 \wedge n \notin N_2 \end{cases}$$

Similarly, the marking for each edge $e \in E$ is specified as:

$$M_E(e) = \begin{cases} 0 & \text{if } e \in E_1 \wedge n \in E_2 \\ 1 & \text{if } e \in E_2 \wedge n \notin E_1 \\ -1 & \text{if } e \in E_1 \wedge n \notin E_2 \end{cases}$$

An example of the difference model is shown in Figure 4(c) and the visual elements of the difference model are specified in the appendix in Table 3. As shown in Figure 4(c), the difference graph uses the color red for deleted nodes and edges and the color green to represent new nodes and edges (i.e. are potential point anomalies). Nodes with white background color represent nodes that exist in both models. It can be seen from Figure 4(c) that structural differences are distinctively observable with this visualization.

Similarity Model.

In addition, we propose the similarity model to visualize the semantic node similarity for each node in the generative RBAC model. Let $RM_1 = (N_1, E_1)$ and $RM_2 = (N_2, E_2)$ be two graph-transformed RBAC models. The similarity model can be specified as:

$$SM := (N_2, E_2, S_N) \tag{6}$$

Hence, we examine the generative RBAC model to identify abnormal semantical changes. The node markings S_N of the similarity model distinguish if a node exists in both models $(sim(n))^2$ or if a node is newly added to RM_2 (t):

$$S_N(n) = \begin{cases} sim(n) & \text{if } n \in N_1 \wedge n \in N_2 \\ t & \text{if } n \in N_2 \wedge n \notin N_1 \end{cases}$$

In our approach, we set the threshold value t to zero. An example of the similarity graph is visualized in Figure 4(d) using the elements listed in the appendix in Table 4. As can be seen from Figure 4(d), the red-colored nodes indicate that the semantic similarity compared to the nodes in the predictive model is very low. Green and orange nodes have a high and medium similarity. Furthermore, blue nodes represent (new) nodes (i.e. they did not exist in the prescriptive model). For example, user u_5 in Figure 4(d) differs highly from its corresponding node in the original model and can be identified as potential contextual anomaly. From the difference graph in Figure 4(c) it can be inferred that this is because of a role change of user u_5 (from role r_3 to r_4). Hence, the combination of both visualizations facilitate the identification and understanding of potential anomalies.

6.3 Experimental Results

In this section, we will evaluate the structural and semantical distance measures with an example data set. Furthermore, we analyze our findings in the context of point and context anomalies.

6.3.1 Data Set

For our evaluation, we analyzed the data sets from [10] provided in *RMiner* [17]. In particular, we used the data sets *firewall2*, *firewall1*, and *university_large* and applied the *HierarchicalMiner* [18] to generate an RBAC model. The results of data sets *firewall1* and *university_large* are not described in this paper due to page limitations however they yield very similar results. For each data set, the resulting RBAC model was converted into a graph-transformed RBAC model and transferred into our prototype.

Based on this original RBAC model RM_1, we injected anomalies to derive RBAC model RM_2 — a "corrupted" version of the original RBAC model. In particular, we injected three types of anomalies: new vertices, missing vertices and connectivity changes (cf. Section 3.1). The algorithms for injecting anomalies into generative RBAC models are outlined in Appendix E. The use of generative models as input is common practice, for example, generative models are produced in Molloy et al. [19]. In the following, we compare the original RBAC model with the corrupted versions based on their unique labels. For example, role r_1 in the original model is matched with the role r_1 in the corrupted version, user u_1 with u_1 and so on. Furthermore, we investigate the impact of the injected anomalies by analyzing the structural and semantic distance of the two RBAC models and sensitivity of the measures to anomalies.

6.3.2 Detecting Anomalies

Example 2. In order to compare RBAC models, we first generated three corrupted (generative) RBAC models of the data set *firewall2* for three types of point anomalies. For point anomaly *new vertices*, we created three corrupted versions where we inserted new nodes (and corresponding edges) that increased the node count by approximately 3.5%, 7%, and 14% of its original vertices. We refer to the resulting RBAC models as $RM_{new,100-p\%}$ where p stands for the percentage increase of vertices (compared to the original model). For the anomaly *missing vertices*, we generated three corrupted RBAC models $RM_{mis,100-p\%}$ by deleting nodes (and their corresponding edges) that approximately account for 3.5%, 7%, and 14% of its vertices. In case of the anomaly *connectivity change*, we created three corrupt versions $RM_{cha,100-p\%}$ where we modified the edge connectivity of the original model in p percent of its edges. In particular, we re-assigned users to roles and permissions to roles. Table 1 shows the key data of the original RBAC model RM_1 and the corrupted models with new vertices ($RM_{new,100-p\%}$), missing vertices ($RM_{mis,100-p\%}$), and connectivity changes ($RM_{cch,100-p\%}$). In addition, it contains the weighted structural complexity (WSC) that sums up the number of relationships in an RBAC state (see Molloy et al. [18] for a specification).

In the following, we will use the models in Example 2 to compute the structural and semantic distance measures and analyze the measures for point and contextual anomalies.

Point Anomalies.

We evaluate point anomalies at two levels. First, we calculate the graph-matching distance measures that indicate the structural difference at *model level*. Secondly, we analyze and visualize point anomalies on *node and edge level* with the difference model as outlined in Section 6.2.

Table 2 displays the distance measures at model level that are represented as bar chart in Figure 5. As can be seen from the figure and table, the more anomalies are injected, the higher is the structural distance to the original model (see d_{mcs} and d_{gu}). Interestingly, in case of new or missing vertices, the distance measures d_{mcs} and d_{gu} are identical. However, in case of changes of the connectivity (see d_{mcs} and d_{gu} in $RM_{cch,100-p\%}$ in Figure 5), the distance based

[2]See Definition 8 in the appendix.

Table 1: Key Information about the Data Sets in Examples 1 and 2

Data set	No. of Nodes			No. of Edges	WSC
	$\|U\|$	$\|R\|$	$\|P\|$		
Fig. 4(a)	9	4	14	29	33
Fig. 4(b)	9	6	13	31	37
RM_1	365	85	720	1312	1397
$RM_{new,96.5}$	384	85	742	1353	1438
$RM_{new,93}$	409	85	759	1395	1480
$RM_{new,86}$	452	85	804	1483	1568
$RM_{mis,96.5}$	337	85	707	1266	1351
$RM_{mis,93}$	320	85	684	1225	1310
$RM_{mis,86}$	290	85	641	1142	1227
$RM_{cch,96.5}$	365	85	720	1312	1397
$RM_{cch,93}$	365	85	720	1312	1397
$RM_{cch,86}$	365	85	720	1312	1397

Table 2: Overview of Distance Measures: Graph Edit Distance d_{ged}, mcs Distances d_{mcs}, Graph Union Distance d_{gu}, Semantic Distance d_{sem} of the Compared RBAC Models $G1$ and $G2$

$G1$	$G2$	d_{ged}	d_{mcs}	d_{gu}	d_{sem}
Fig. 4(a)	Fig. 4(b)	31	0.28814	0.42466	0.48666
RM_1	$RM_{new,96.5}$	82	0.03198	0.03198	0.04934
RM_1	$RM_{new,93}$	166	0.06269	0.06269	0.09361
RM_1	$RM_{new,86}$	342	0.12111	0.12111	0.17742
RM_1	$RM_{mis,96.5}$	87	0.03505	0.03505	0.04833
RM_1	$RM_{mis,93}$	168	0.06769	0.06769	0.09624
RM_1	$RM_{mis,86}$	324	0.13054	0.13054	0.17799
RM_1	$RM_{cch,96.5}$	76	0.01531	0.03016	0.05000
RM_1	$RM_{cch,93}$	156	0.03143	0.06094	0.09453
RM_1	$RM_{cch,86}$	302	0.06084	0.11470	0.30928

on the idea of graph union d_{gu} increases stronger than the distance based on the maximum common subgraph d_{mcs}. Accordingly, we can convey that the change of vertices of $p\%$ also increases the distance between the models to approximately $p\%$. However, only in the case of reassignments of edges ($RM_{cch,100-p\%}$), the distance d_{mcs} increases $\frac{p}{2}\%$.

Hence, structural distance measures indicate the structural modifications between RBAC models. However, they do not give information on the semantical changes. In the following, we will investigate these (contextual) changes.

Contextual Anomalies.

The identification and investigation of contextual anomalies is challenging as contextual anomalies are influenced by point anomalies (e.g., new vertices or changes in the connectivity of edges). In our approach, we evaluate the contextual anomalies at two levels. First, we evaluate the semantic similarity on *model level* by computing the similarity measure between two RBAC models. Secondly, we investigate contextual anomalies on *node level* by analyzing the semantic similarity for each node in a graph-transformed RBAC model. Furthermore, we visualize the node similarity in the similarity model (see Section 6.2).

The results of the semantic distance measure d_{sem} is shown in Table 2 and Figure 5. It can be seen from the figure that structural modifications can lead to semantic changes in the composition of the RBAC model. We can infer that the more anomalies are injected, the higher is their semantic distance. In Example 2, the reassignment of 14% of edges (i.e. user-role and role-permission assignments) leads to a steep increase in the semantic distance (see d_{sem} for

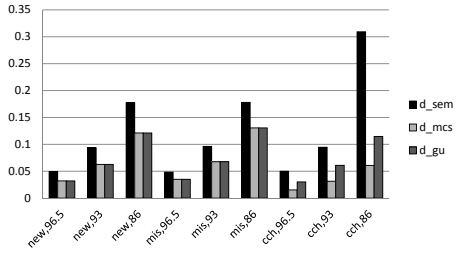

Figure 5: Example2: Distance Measures of Corrupted RBAC Versions compared to the Original RBAC Model

$RM_{cch,86}$ in Figure 5). By analyzing the difference graph $DM(RM_1, RM_{cch,86})$, we identified that $RM_{cch,86}$ contains many user-role and role-permission assignments that were in the original model RM_1 connected to roles in the upper hierarchy and are in $RM_{cch,86}$ associated with roles in the lower hierarchy. We discovered similar effects in the data sets *university_large* and *firewall1*. Subsequently, modifications in the upper hierarchy have a strong impact on the semantic similarity. We will evaluate the impact of changes in upper hierarchy levels in future research.

When comparing the measures with the WSC, it can be seen that structural changes can increase or decrease the WSC (see Table 1). However, the metric is limited when RBAC changes are injected and the WSC of the prescriptive and generative model is identical (see models $RM_{cch,100-p\%}$). In this case, the structural and semantical RBAC changes (e.g., in $RM_{cch,86}$ or $RM_{cch,93}$) cannot be exposed as WSC summarizes the RBAC state.

7. RELATED WORK

Anomaly detection is an important problem that has been researched in many areas of application domains such as intrusion detection or statistics (see [6]). Furthermore, graphs can be used for anomaly detection such as in Noble et al. [20] where anomalous substructures or subgraphs are detected. In case of RBAC, Park et al. [21] provide a role-oriented profile analysis in order to monitor insiders using frequency patterns. In this paper, we use graph-transformed RBAC models to analyze the behavior of insiders by analyzing structural and semantic differences.

Generative RBAC models have gained increased attention from the research community (e.g., [19, 3, 7]). For example in Baumgrass et al. [3], model matching techniques are used to migrate a generative RBAC model to a target-state RBAC model. Here, the migration is performed via edit operations to transform one model to another. Our approach aims to investigate anomalies between RBAC models to identify insider threats. Therefore, we use edit operations to compute the distance measures between graphs. Another example for RBAC comparison is the delta analysis in Leitner [15] that aims to compare two RBAC models in order to detect errors or violations in RBAC configurations. In this paper, we propose structural and semantic measures that can be used for delta analysis.

Moreover, graph representations of RBAC models are used such as in Koch et al. [12] that provide a formalization of RBAC models using graph transformations. Another example in Zhang et al. [29] uses RBAC graphs for role engineering. Furthermore, research has provided several metrics to analyze characteristics of RBAC such as the WSC in Molloy et al. [18] or the accuracy or coverage of roles

in Zhang et al. [27]. For example, the accuracy and coverage are metrics to identify similarities for role sets and user-permission sets. In our approach, we investigate similarities on node (user, role, and permission) and model level. Moreover, [24, 23] measure the similarity between two role sets with the Jaccard coefficient. We adapt this approach to evaluate the semantic similarity between roles.

8. CONCLUSION

This paper provided an approach to identify anomalies in RBAC models that may indicate insider threats. Thereby, we compare a prescriptive RBAC model to a generative RBAC model (e.g., derived from event logs). Furthermore, we provide metrics for structural and semantic differences between RBAC models. In addition, we provide visualization techniques to evaluate our metrics with an experimental setup and examples. Our approach not only identifies and exposes potential anomalies in RBAC models but also can be used for RBAC alignments. As this was an initial step towards anomaly detection in RBAC models, for future work, we plan to investigate semi-supervised techniques and to discover collective anomalies.

9. ACKNOWLEDGMENTS

The research was funded by COMET K1, FFG - Austrian Research Promotion Agency.

10. REFERENCES

[1] M. Alanen and I. Porres. Difference and union of models. In *UML '03*, number 2863 in LNCS, pages 2–17. Springer, Jan. 2003.

[2] A. Baumgrass. Deriving current-state RBAC models from event logs. In *ARES '11*, pages 667–672, 2011.

[3] A. Baumgrass and M. Strembeck. Bridging the gap between role mining and role engineering via migration guides. *Information Security Technical Report*, 17(4):148–172, May 2013.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[5] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3–4):255–259, Mar. 1998.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[7] S. Chari, I. Molloy, Y. Park, and W. Teiken. Ensuring continuous compliance through reconciling policy with usage. In *SACMAT '13*, page 49–60. ACM, 2013.

[8] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, May 2004.

[9] P. J. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7(3):243–254, Dec. 2004.

[10] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *SACMAT '08*, page 1–10. ACM, 2008.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[12] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A graph-based formalism for RBAC. *ACM Trans. Inf. Syst. Secur.*, 5(3):332–365, Aug. 2002.

[13] S. Kriglstein, G. Wallner, and S. Rinderle-Ma. A visualization approach for difference analysis of process models and instance traffic. In *BPM*, pages 219–226. Springer, 2013.

[14] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03*, page 179–186. ACM, 2003.

[15] M. Leitner. Delta analysis of role-based access control models. In *EUROCAST '13*, volume 8111 of *LNCS*, pages 507–514. Springer, 2013.

[16] M. Leitner and S. Rinderle-Ma. A systematic review on security in process-aware information systems – constitution, challenges, and future directions. *Information and Software Technology*, 56:273–293, 2014.

[17] R. Li, H. Li, W. Wang, X. Ma, and X. Gu. RMiner: a tool set for role mining. In *SACMAT '13*, page 193–196. ACM, 2013.

[18] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. In *SACMAT '08*, page 21–30. ACM, 2008.

[19] I. Molloy, Y. Park, and S. Chari. Generative models for access control policies: applications to role mining over logs with attribution. In *SACMAT '12*, page 45–56. ACM, 2012.

[20] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD '03*, page 631–636. ACM, 2003.

[21] J. Park and J. Giordano. Role-based profile analysis for scalable and accurate insider-anomaly detection. In *IPCCC '06*, 2006.

[22] M. Strembeck. Scenario-driven role engineering. *Security Privacy, IEEE*, 8(1):28 –35, Feb. 2010.

[23] H. Takabi and J. B. Joshi. StateMiner: an efficient similarity-based approach for optimal mining of role hierarchy. In *SACMAT '10*, page 55–64. ACM, 2010.

[24] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. In *SACMAT '07*, page 175–184. ACM, 2007.

[25] J. Vaidya, V. Atluri, Q. Guo, and N. Adam. Migrating to optimal RBAC with minimal perturbation. In *SACMAT '08*, page 11–20. ACM, 2008.

[26] W. D. Wallis, P. Shoubridge, M. Kraetz, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6–7):701–704, May 2001.

[27] D. Zhang, K. Ramamohanarao, T. Ebringer, and T. Yann. Permission set mining: Discovering practical and useful roles. In *ACSAC '08*, pages 247–256, 2008.

[28] D. Zhang, K. Ramamohanarao, S. Versteeg, and R. Zhang. RoleVAT: visual assessment of practical need for role based access control. In *ACSAC '09*, pages 13–22, 2009.

[29] D. Zhang, K. Ramamohanarao, S. Versteeg, and R. Zhang. Graph based strategies to role engineering. In *CSIIRW '10*, page 25:1–25:4. ACM, 2010.

APPENDIX

A. REVIEW FUNCTIONS

The review functions required in this paper are defined as follows (based on [11]):

Definition 7 (Review Functions on RBAC Models). *Let $RBAC := (U, R, P, UA, PA, RH)$ be an RBAC model. Then we define the following review functions:*

- $\texttt{d\_senior} : R \mapsto 2^R$ *with* $\texttt{d\_senior}(r) = \{r' \in R \mid \exists (r', r) \in RH\}$
- $\texttt{senior} : R \mapsto 2^R$ *with* $\texttt{senior}(r) = \{r' \mid r' \in \texttt{d\_senior}(r) \vee \exists r'' \in \texttt{d\_senior}(r) : r' \in \texttt{senior}(r'')\}$
- $\texttt{d\_junior} : R \mapsto 2^R$ *with* $\texttt{d\_junior}(r) = \{r' \in R \mid \exists (r, r') \in RH\}$
- $\texttt{junior} : R \mapsto 2^R$ *with* $\texttt{junior}(r) = \{r' \mid r' \in \texttt{d\_junior}(r) \vee \exists r'' \in \texttt{d\_junior}(r) : r' \in \texttt{junior}(r'')\}$
- $\texttt{authorizedUsers} : R \mapsto 2^U$ *with* $\texttt{authorizedUsers}(r) = \{u \in U \mid \exists ua \in UA \text{ with } ua = (u, r) \vee ua = (u, \texttt{senior}(r))\}$
- $\texttt{authorizedRoles} : U \mapsto 2^R$ *with* $\texttt{authorizedRoles}(u) = \{r \in R \mid \exists ua \in UA \text{ with } ua = (u, r) \vee ua = (u, \texttt{junior}(r))\}$
- $\texttt{userPermissions} : U \mapsto 2^P$ *with* $\texttt{userPermissions}(u) = \{p \in P \mid \exists ua \in UA, pa \in PA \text{ with } (ua = (u, r) \wedge pa = (r, p)) \vee (ua = (u, \texttt{junior}(r)) \wedge pa = (\texttt{junior}(r), p))\}$
- $\texttt{permissionsUser} : P \mapsto 2^U$ *with* $\texttt{permissionsUser}(p) = \{u \in U \mid \exists ua \in UA, pa \in PA \text{ with } (ua = (u, r) \wedge pa = (r, p)) \vee (ua = (u, \texttt{senior}(r)) \wedge pa = (\texttt{senior}(r), p))\}$
- $\texttt{rolePermissions} : R \mapsto 2^P$ *with* $\texttt{rolePermissions}(r) = \{p \in P \mid \exists pa \in PA \text{ with } pa = (r, p) \vee pa = (\texttt{junior}(r), p)\}$
- $\texttt{permissionRoles} : P \mapsto 2^R$ *with* $\texttt{permissionRoles}(p) = \{r \in R \mid \exists pa \in PA \text{ with } pa = (r, p) \vee pa = (\texttt{senior}(r), p)\}$

B. SIMILARITY FUNCTION

For the computation of the semantic similarity, the similarity function is defined as follows:

Definition 8 (Similarity Function). *The similarity function $sim(n)$ returns the node similarity for a node n.*

$$sim(n) = \begin{cases} userSim(u_i, u_j) \text{ for } n = u_j \\ roleSim(r_i, r_j) \text{ for } n = r_j \\ permSim(p_i, p_j) \text{ for } n = p_j \end{cases} \quad (7)$$

C. ALGORITHM SEMANTIC DISTANCE

Algorithm 1 computes the semantic distance between two RBAC models.

Algorithm 1 Semantic Distance: dSem(RM_1, RM_2)

Require: Two RBAC models: RM_1, RM_2.
Require: Weights $w_{ur}, w_{up}, w_{ru}, w_{rh}, w_{rp}, w_{sen}, w_{jun}, w_{pu}, w_{pr}$.
Require: Threshold t.

$sumSemSim \leftarrow 0$
$count \leftarrow 0$
$nodeSim \leftarrow 0$
$matchedNodes = N_1 \cap N_2$
for each $n \in N_1 \cup N_2$ **do**
  if $n \in matchedNodes$ **then**
    if n is a user node **then**
      $simUR \leftarrow \frac{|\texttt{authorizedRoles}(n_1) \cap \texttt{authorizedRoles}(n_2)|}{|\texttt{authorizedRoles}(n_1) \cup \texttt{authorizedRoles}(n_2)|}$
      $simUP \leftarrow \frac{|\texttt{userPermissions}(n_1) \cap \texttt{userPermissions}(n_2)|}{|\texttt{userPermissions}(n_1) \cup \texttt{userPermissions}(n_2)|}$
      $nodeSim \leftarrow w_{ur} * simUR + w_{up} * simUP$
    else if n is a role node **then**
      $simRU \leftarrow \frac{|\texttt{authorizedUsers}(n_1) \cap \texttt{authorizedUsers}(n_2)|}{|\texttt{authorizedUsers}(n_1) \cup \texttt{authorizedUsers}(n_2)|}$
      $simRH \leftarrow w_{sen} * \frac{min(|\texttt{senior}(n_1)|, |\texttt{senior}(n_2)|)}{max(|\texttt{senior}(n_1)|, |\texttt{senior}(n_2)|)} + w_{jun} * \frac{min(|\texttt{junior}(n_1)|, |\texttt{junior}(n_2)|)}{max(|\texttt{junior}(n_1)|, |\texttt{junior}(n_2)|)}$
      $simRP \leftarrow \frac{|\texttt{rolePermissions}(n_1) \cap \texttt{rolePermissions}(n_2)|}{|\texttt{rolePermissions}(n_1) \cup \texttt{rolePermissions}(n_2)|}$
      $nodeSim \leftarrow w_{ru} * simRU + w_{rh} * simRH + w_{rp} * simRP$
    else if n is a permission node **then**
      $simRP \leftarrow \frac{|\texttt{permissionRoles}(n_1) \cap \texttt{permissionRoles}(n_2)|}{|\texttt{permissionRoles}(n_1) \cup \texttt{permissionRoles}(n_2)|}$
      $simPU \leftarrow \frac{|\texttt{permissionUsers}(n_1) \cap \texttt{permissionUsers}(n_2)|}{|\texttt{permissionUsers}(n_1) \cup \texttt{permissionUsers}(n_2)|}$
      $nodeSim \leftarrow w_{pr} * simRP + w_{pu} * simPU$
    end if
    $sumSemSim \leftarrow sumSemSim + nodeSim$
    $count \leftarrow count + 1$
    $nodeSim \leftarrow 0$
  else
    $nodeSim \leftarrow t$
    $sumSemSim \leftarrow sumSemSim + nodeSim$
    $count \leftarrow count + 1$
  end if
end for
return $\left(1 - \frac{sumSemSim}{count}\right)$

D. VISUALIZATION

The visualization of the difference graph requires the definition of the visual elements as shown in Table 3. Furthermore, Table 4 outlines the visual elements used in the similarity graph. See Appendix B for a definition of the function *sim*. For x, y, we specify $0.0 < x < y < 1.0$.

E. ALGORITHMS FOR ANOMALY INJECTION

Algorithm 2 injects new vertices and Algorithm 3 injects "missing" vertices into a graph-based representation of an RBAC model. Algorithm 4 injects connectivity changes into an RBAC model; it reassigns user or permission nodes to role nodes. In the following, we describe the utilized functions in the algorithms:

- $createRandomUserPermVertex()$ creates a new user or permission node,
- $selectRandomEdge(E)$ returns a random edge from a edge set E,
- $selectRandomVertex(N)$ returns a random vertex from a node set N,
- $selectRandomVertexSet(N)$ returns a random set of vertices from a node set N,
- $source()$ returns the source node and $target()$ returns the target node of a directed edge,
- and $outNeighbors()$ returns a set of out-neighbors of a vertex, and $inNeighbors()$ returns a set of in-neighbors of a vertex.

Table 4: Representation of Nodes/ Edges in the Similarity Model

Symbol	Meaning	Description
Green node	High similarity	$\forall n \in N_2 \wedge n \in N_1$ where $sim(n) \geq y \wedge sim(n) \leq 1.0$
Orange node	Medium similarity	$\forall n \in N_2 \wedge n \in N_1$ where $sim(n) \geq x \wedge sim(n) < y$
Red node	Low similarity	$\forall n \in N_2 \wedge n \in N_1$ where $sim(n) \geq 0.0 \wedge sim(n) < x$
Blue node	New node	$\forall n \in N_2 \wedge n \notin N_1$ where $sim(n) = t$

Table 3: Representation of Nodes/ Edges in the Difference Model

Symbol	Meaning	Description
	New node	$\forall n \in N_2$ for $M_N(n) = 1$
	Deleted node	$\forall n \in N_2$ for $M_N(n) = -1$
	New edge	$\forall e \in E_2$ for $M_E(n) = 1$
	Deleted edge	$\forall e \in E_2$ for $M_E(n) = -1$

Algorithm 2 Injection of new vertices: $injectNewVertices(RM, p)$

Require: RBAC model: $RM = (U, R, P, UA, PA, RH)$.
Require: Percentage p.

$numNewVertices \leftarrow \frac{|U|+|R|+|P|}{100} * p$
for $i = 1$ to $numNewVertices$ **do**
  $n \leftarrow createRandomUserPermVertex()$
  $r \leftarrow selectRandomVertex(R)$
  if n is a user node **then**
    $U \leftarrow U \cup \{n\}$
    $RS \leftarrow selectRandomVertexSet(R)$
    for each $r \in RS$ **do**
      $UA \leftarrow UA \cup \{(n, r)\}$
    end for
  else if n is a permission node **then**
    $P \leftarrow P \cup \{n\}$
    $RS \leftarrow selectRandomVertexSet(R)$
    for each $r \in RS$ **do**
      $PA \leftarrow PA \cup \{(r, n)\}$
    end for
  end if
end for
return RM

Algorithm 3 Injection of "missing" vertices: $injectMisVertices(RM, p)$

Require: RBAC model: $RM = (U, R, P, UA, PA, RH)$.
Require: Percentage p.

$numMisVertices \leftarrow \frac{|U|+|R|+|P|}{100} * p$
for $i = 1$ to $numMisVertices$ **do**
  $n \leftarrow selectRandomVertex(U \cup P)$
  if n is a user node **then**
    for each $(n, r) \in UA$ **do**
      $UA \leftarrow UA \setminus \{(n, r)\}$
    end for
    $U \leftarrow U \setminus \{n\}$
  else if n is a permission node **then**
    for each $(r, n) \in PA$ **do**
      $PA \leftarrow PA \setminus \{(r, n)\}$
    end for
    $P \leftarrow P \setminus \{n\}$
  end if
end for
return RM

Algorithm 4 Injection of connectivity changes: $injectCChanges(RM, p)$

Require: RBAC model: $RM = (U, R, P, UA, PA, RH)$.
Require: Percentage p.

$numCchEdges \leftarrow \frac{|U|+|R|+|P|}{100} * p$
for $i = 1$ to $numCchEdges$ **do**
  $e \leftarrow selectRandomEdge(UA \cup PA)$
  $sourceVertex \leftarrow e.source()$
  $targetVertex \leftarrow e.target()$
  if $e \in UA$ **then**
    $RS \leftarrow R \setminus \{targetVertex\}$
    $RS \leftarrow RS \setminus \{sourceVertex.outNeighbors()\}$
    $r \leftarrow selectRandomVertex(RS)$
    $UA \leftarrow UA \setminus \{e\}$
    $UA \leftarrow UA \cup \{(sourceVertex, r)\}$
  else if $e \in PA$ **then**
    $RS \leftarrow R \setminus \{sourceVertex\}$
    $RS \leftarrow RS \setminus \{targetVertex.inNeighbors()\}$
    $r \leftarrow selectRandomVertex(RS)$
    $PA \leftarrow PA \setminus \{e\}$
    $PA \leftarrow PA \cup \{(r, targetVertex)\}$
  end if
end for
return RM