Identification and Attribution of Access Roles Using Hierarchical Team Permission Analysis

Iryna Didinova[1][a] and Karel Macek[1,2][b]
[1]*AI Center of Excellence, Generali Ceska Pojistovna, Na Pankraci 1720, Prague, Czech Republic*
[2]*AI & Data Center of Excellence, Temus, 80 Pasir Panjang Road, Singapore*

Keywords: RBAC, Machine Learning, Clustering, Attribution, Graph Algorithms, Access Control Models.

Abstract: This paper addresses the challenges of Role-Based Access Control (RBAC) in large organizations, with a focus on the efficient attribution of access roles. It critiques traditional role-mining algorithms and the use of Machine Learning (ML) models, which serve as benchmarks due to their lack of practical interpretability and potential security vulnerabilities. The novel contribution of this work is the introduction of the Hierarchical Team Permission Analysis (HTPA), a methodology grounded in organizational hierarchy. HTPA is shown to outperform the benchmark approaches by creating meaningful, interpretable roles that enhance both the security and efficiency of access control systems in large enterprises. The paper advocates for the potential integration of HTPA with ML models to further optimize role attribution and suggests avenues for future research in this evolving field.

1 INTRODUCTION

Access control helps protect sensitive information like customer data and intellectual property from theft by hackers or unauthorized access. Therefore, it is crucial for ensuring data privacy. Access roles are often not available due to system attribute limitations, inconsistent access, the impact of mergers, and overly granular role definitions, based on reflective analysis. Efficient IT operation in large organizations requires a reasonable access management that combines security and protection on one hand, as well as ease of use and efficiency on the other hand (Di Pietro et al., 2012). Additionally, the growing interest in Artificial Intelligence and Big Data, which often includes personal information, sparks fresh attention in how access controls are applied and developed (Cavoukian et al., 2015).

Many organizations manage the permissions assignment inefficiently by assigning them directly to the users. A much better and more practical approach is to group permissions to roles, and assign those roles to users, which is the so-called RBAC (Role-Based Access Control) model (Sandhu et al., 1996).

This paper addresses a problem that has two components: i. to identify the RBAC roles, and ii. to have mechanisms of assigning (attributing) these roles to users based on some metadata efficiently. Such approach would be a combination of RBAC and attribute-based access control (ABAC) (Karp et al., 2010). This combination has potential to bring the best from both access control models (Alayda et al., 2020) and is particularly useful for large organizations (Kuhn et al., 2010).

2 LITERATURE REVIEW

Our research on role identification highlights extensive studies, including algorithms like *CompleteMiner* and *FastMiner*, which proceed through candidate role generation from user-permission data, followed by prioritizing these roles to produce a prioritized list (Vaidya et al., 2006). Another notable method is the Graph Optimization (GO) algorithm, which initially considers each user's permissions as separate roles and then optimizes by merging or splitting roles based on an optimization policy (Zhang et al., 2007).

Some role-mining algorithms treat the process as an optimization problem, focusing on minimizing criteria like the Weighted Structural Complexity (WSC)

[a] https://orcid.org/0009-0001-8722-7712
[b] https://orcid.org/0000-0002-3914-447X

Didinova, I. and Macek, K.
Identification and Attribution of Access Roles Using Hierarchical Team Permission Analysis.
DOI: 10.5220/0012675300003690
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 721-728
ISBN: 978-989-758-692-7; ISSN: 2184-4992
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

(Molloy et al., 2008). WSC accounts for various factors such as the number of roles and errors, as well as the assignments between users and roles, roles and permissions, and the role hierarchy. It also allows for different components of WSC to be weighted differently. Molloy et al. developed the HierarchicalMiner algorithm to specifically minimize the total WSC (Molloy et al., 2008).

The algorithms mentioned above output a mathematically optimized set of roles. Nevertheless, such roles often lack a reasonable interpretation. In practice, the generated roles are often examined by a security administrator, who decides whether these roles are meaningful and whether they should be implemented. As a result, roles often stay unused in practice due to this problem: the stakeholders do not want to deploy roles they can't understand (Ene et al., 2008). Furthermore, the algorithms mentioned above neither take into consideration users' attributes nor allow for assigning users to roles, and due to this issue, neither of the algorithms is well-suited for the problem of this paper.

Another possibility is Rule-Based RBAC, which involves experts creating rules for assigning roles to users, which are then automated (Al-Kahtani and Sandhu, 2004). However, this method can be costly and time-consuming due to the need for detailed analysis of user-role relations, leading to only partial automation (Ni et al., 2009).

Machine learning (ML) and Artificial Intelligence (AI) are increasingly applied to enhance access control systems, addressing role attribution challenges. For example, Lu Zhou et al. introduced an ML approach for real-time role assignment in SCADA systems, employing Support Vector Machines (SVM) to identify roles and Adaboost for classification based on static and dynamic user attributes (Zhou et al., 2019).

Nobi et al. introduced DLBAC, a Deep Learning Based Access Control system (Nobi et al., 2022b), addressing limitations of standard RBAC such as role attribution and maintenance issues. DLBAC employs a neural network that takes user and resource metadata as input to make access decisions. The network's classification layer outputs probabilities for granting permissions, with each neuron representing an operation. In practice, when the network predicts a probability of 1, access is granted upon user request.

With recent advances in Generative AI, there's now talk about its potential to automate the creation of policies for ABAC (Olabanji et al., 2024). Jayasundara et al. go even further and introduces "AccessFormer"- a feedback-driven access control policy generation framework. Authors use Large Language Models (LLMs) to generate policies from high-level requirement specifications, which are refined by security administrator feedback (Jayasundara et al., 2024).

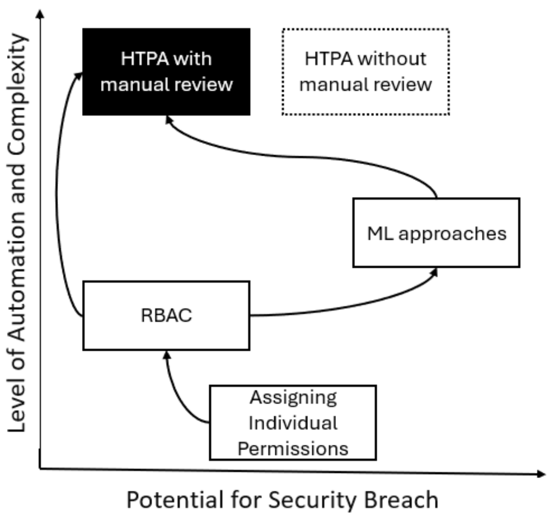
HTPA with manual review
HTPA without manual review
ML approaches
RBAC
Assigning Individual Permissions
Level of Automation and Complexity
Potential for Security Breach

Figure 1: Hierarchical Team Permission Analysis (black box) in the context of other prior efforts.

The results of the ML methods in access control are promising. Nevertheless, there are some challenges. The first one is a lack of understandability and interpretation in results produced by non-symbolic models (NN, SVM, Random forest, Clustering). Another pitfall of ML is a need for quality, sizable data sets with extensive user and policy attributes, and access control data sets often do not meet these requirements (Nobi et al., 2022a). Lastly, using ML models in access control could be risky due to their probabilistic nature and some level of inaccuracy. In case model mistakenly assigns a role or permission to a wrong person, it could lead to security breaches and high loses for an enterprise.

Due to such drawbacks, neither the above-mentioned conventional RBAC algorithms nor ML methods are well suited for our research question. As a result, we introduce our solution, a graph algorithm based on companies' hierarchical structure: Hierarchical Team Permission Analysis. It is an evolutionary step from the application of ML techniques in the combination with attribution, as visually represented in Figure 1 as a black box. It is important to mention that the method assumes manual review of the proposed roles (by the line managers). Without that, the risk of potential security breach would be high, as depicted in Figure 1 as a dashed box.

The paper is organized as follows. In Section 3, we define the problem statement, particularly we specify the business requirements, introduce a notation as well as an evaluation criterion. In Section 4 we propose two approaches to find the solution, particularly the ML approach and Hierarchical Team Permission Analysis. Section 5 showcases the algorithms in a real-world case. Finally, Section 6 concludes the paper and outlines possible areas for future research.

3 PROBLEM STATEMENT

3.1 Business Requirements

To better grasp the issue at hand, it's crucial to specify the main business needs.

The business objective is to identify accurate roles that can cover most of the initial user-permission assignments while reducing the instances of incorrectly given permissions. Thus, we encounter the challenge of balancing two types of errors: the false negative rate (failing to grant permissions to a user who actually needs them) and the false positive rate (inaccurately granting permissions to users who shouldn't have them). A high false negative rate implies additional work for security administrators, who must then manually allocate the missing permissions. Conversely, a high false positive rate could lead to security vulnerabilities. These two error types could be balanced during the optimization phase by applying different weights.

Nevertheless, we need to account for the possibility of false positives, although we acknowledge that they are less desirable than false negatives. There may be certain user-permission pairs that do not pose any risk and may simplify the roles. However, these false positive cases should be checked manually by line managers or security professionals.

Another business requirements is, that roles should be attributed based on metadata. This implies we aim to understand not just the roles users hold but also the reasons behind them, based on user information. This requirement is closely related to the first one as slightly relaxed roles with controlled false positives can provide data that can be better generalized by attribution rules. This is meant to ensure that each person is assigned a role that aligns with their competencies and position in the organization, balancing the need for functional access with the principle of least privilege.

Finally, the last business requirement is, that roles should have a reasonable business interpretation.

3.2 Notation

We adopt the notation from (Molloy et al., 2008) with a slight modification. Cardinality of a set is denoted as $|\cdot|$, composition of relations as $R_1 \circ R_2$

Given an access control configuration is a tuple $\langle U,P,UP \rangle$, where U is a set of users, P is a set of permissions, and $UP \subseteq U \times P$ is the user-permission assignment.

An RBAC policy is a tuple $\langle U,P,R,UA,PA,RH,DUPA,NUPA \rangle$, where R is a set of roles, $UA \subseteq U \times R$ is the user-role assignment, $PA \subseteq R \times P$ is the permission-role assignment, and $RH \subseteq R \times R$ is a partial order over R, which is called a role hierarchy, and $DUPA \subseteq U \times P$ is the direct user-permission assignment relation. In machine learning, it can also be interpreted as false negative cases. In addition to the original definition of WSC, we would like to introduce a metric $NUPA \subseteq U \times P$, which is the direct corrective unassigments (or false positive cases).

The RBAC state is consistent with $\langle U,P,UP \rangle$, if every user in U has the same set of authorized permissions in the RBAC state as in UP.

Additionally, we will consider attributes for each user $A : U \to \mathbb{R}^n$.

We introduce a structured set of teams, T, organized hierarchically as a tree $H \subseteq T \times T$. Within this hierarchy, a pair $(t,t') \in H$ signifies that team t is directly subordinate to a team t'. To represent user membership across these teams, we employ the mapping function $M : U \to 2^T$. For each user u within the set U, the function outputs a subset of teams $M(u) \subseteq T$ that encapsulates the user's memberships. It is assumed that the teams within $M(u)$ form a path in H, indicating that a user is affiliated with a distinct team at the most specific level of the hierarchy (denoted as $\mu(u)$, $\mu : U \to T$) and all its ancestor teams up to the root of the tree (denoted as $t_0 \in T$). Note that the "most specific"team $\mu(u)$ does not have to be a leaf in the hierarchy; it can be an internal node representing a team with further subdivisions or subteams. The user might be a manager (or her secretary) of that team and all the subteams.

For better readability, we summarize all the used symbols in Table 1, including those that will be defined later in the text:

3.3 Tasks

For reasons summarized in Section 3.1, we split - in contrast to the algorithms described in the literature - the RBAC creation problem into two tasks.

3.3.1 Role Mining

The first task is identifying the roles R, their permissions PA, and their hierarchy RH.

Table 1: Table of Symbols and Their Meanings.

Symbol	Meaning
U	Set of users
P	Set of permissions
UP	User-permission assignment
R	Set of roles
UA	User-role assignement
PA	Permission-role assignement
RH	Role hierarchy
$DUPA$	Direct corrective assignments
$NUPA$	Direct corrective unassignments
T	Set of teams
H	Hierarchy tree of teams
M	User memberships in teams
$\mu(u)$	Most specific team of user u
$A(u)$	Attributes of user u
n	Dimension of vector $A(u)$
F_r	Attribution rule for role r
$\overline{UP}$	Binary matrix representing UP
$\overline{PA}$	Binary matrix representing PA
$\overline{UA}$	Binary matrix representing UA
$\overline{Q}$	Real matrix - output from clustering
η	Threshold for binarization $\overline{Q}$ to $\overline{PA}$
m	Minimum number of role's users
λ	Threshold for user-role assignment
θ	Threshold percentage for defining common permissions in HTPA
$\Pi(t')$	Permissions common for team t'
$\omega(t',p)$	Ratio of users in team t' having permission p

3.3.2 Attribution

The second task is attributing the roles to users, using attribution rules. Instead of a direct construction of UA, we create a decision rule $F_r : \mathbb{R}^n \rightarrow \{0,1\}$ for each role $r \in R$ that decides on the assignment to user $u \in U$ based on their attributes $A(u)$. In the ML language, we can interpret F_r as a classification model.

3.4 Evaluation

The task is to find an optimal and attributable RBAC policy for a given access control configuration so the overall criterion on the test data set is minimized.

$$\begin{aligned} wsc(\gamma, W) = & \; w_r \cdot |R| \\ & + w_u \cdot |UA| \\ & + w_p \cdot |PA| \\ & + w_h \cdot |t_{reduce}(RH)| \\ & + w_d \cdot |DUPA| \\ & + w_n \cdot |NUPA| \end{aligned} \tag{1}$$

This criterion is called Weighted Structural Complexity (Molloy et al., 2008).

Moreover, we will use a secondary metric to evaluate the roles: a Covering rate. In machine learning terminology, this can be considered the true-positive rate. The Covering rate is calculated as:

$$CR = \frac{|UP \cap UA \circ PA|}{|UP|}$$

4 SOLUTION

In our research, two main approaches were tried: the standard ML approach, particularly dimensionality reduction together with clustering, and Hierarchical Team Permission Analysis (HTPA), based on the hierarchical structure of the organization.

4.1 Description of Data

Input UP is represented by a sparse binary matrix $\overline{UP}$, having rows as users and columns as permissions. Such conversion is motivated by the application of ML algorithms as well as HTPA algorithm, since both of them require matrix inputs.

The hierarchical structure of the organization is represented by a table, where first column represents a child node, particularly User ID or Organization Unit ID, and the second column is the parent node. From this data we are able to generate a hierarchy tree, starting from the top level, which is the whole company, and moving to the bottom level, which is the user's team. For example: Company - IT department - Data Science team.

The desired data output would be: 1) PA represented as a binary $\overline{PA}$ matrix, where rows are roles and columns are permissions; 2) UA represented as a binary $\overline{UA}$ matrix, where rows are users and columns are roles.

4.2 Machine Learning Approach

4.2.1 Clustering

There have been several attempts to apply clustering techniques for the role-mining. Kuhlmann and Kern present a clustering technique similar to the widely-known k-means clustering, which requires predefining the number of clusters (Kern et al., 2002). Another prominent work in this field was ORCA method based on the hierarchical clustering algorithm to form roles and then illustrate the role hierarchy in a graphical form (Schlegelmilch and Steffens, 2005). Abolfathi et al. (2021) suggested a highly scalable solution to the role-mining problem, suitable for the large organizations. Non-negative matrix factorization was used, which is a dimensionality reduction technique and is close to the clustering methods (Abolfathi et al., 2021).

Based on the above-mentioned literature, we decided to use a combination of a dimensionality reduction technique and clustering for the role-mining, which should be applied to the sparse matrix $\overline{UP}$. It is well known, that the clustering algorithms are sensitive to the so-called "curse of dimensionality"(Köppen, 2000), and so do not perform well in high dimensional space (Assent, 2012). Moreover, multiple permissions could appear together and thus be highly correlated. Due to this fact we first applied a dimensionality reduction technique to the UP matrix, particularly - linear dimensionality reduction using truncated singular value decomposition (SVD). Contrary to principal component analysis (PCA), this estimator does not center the data before computing the singular value decomposition, and hence it can work with sparse matrices efficiently (Schütze et al., 2008)

After that, k-means clustering (Lloyd, 1982) is applied to the new low-dimensional space to find centroids of clusters. Subsequently, these centroids are decomposed to permissions, which result in a non-binary matrix $\overline{Q}$. Then, this matrix is binarized to $\overline{PA}$ that corresponds to PA, using a threshold η. To find an optimal value of η, the WSC (1) is optimized.

As a result the roles R and permission assignments PA (represented as a binary $\overline{PA}$ matrix) is obtained.

4.2.2 Attribution of Roles Using Logistic Regression

According to the business requirements in Section 3.1, the roles R should be assigned to users U, i.e., UA should be constructed based on the attribution rules. We constructed them as classifiers, adopting logistic regression as a well-interpretable model. Each role will have one classifier.

Firstly, the data for machine learning are constructed using users' attributes $X = [A(u)]_{u \in U}$ as an input. For each user and each role, we test if user $u \in U$ has a role $r \in R$ - this serves as output Y. The role r is assigned to the user u, if she/he has relatively at least λ permissions in this role.

$$\frac{|p : p \in P, (r, p) \in PA, (u, p) \in UP|}{|p : p \in P, (r, p) \in PA|} \geq \lambda$$

The parameter λ, slightly less than 1, was introduced to enhance the algorithm's adaptability. Its implementation permits some degree of $NUPA$, while facilitating more effective minimization of WSC.

In this logic, a vector Y is built for each role. Moreover, the roles having less than m users were removed because it is impossible to train a meaningful logit model on a small number of observations.

Using this approach, multiple logistic regression models F_r for $r \in R$ were created to predict (attribute) every role to users. This prediction (not the original values in Y) was used for as UA, i.e.

$$(u, r) \in UA \text{ if and only if } F_r(A(u)) = 1$$

4.3 Hierarchical Team Permission Analysis

Processing the hierarchy of teams, considering standard processing of a tree, like DFS or BFS. The buffer contains pairs of (team id, considered permissions).

1. Place the top hierarchy (whole company) into the buffer, with all permission columns, i.e., (t_0, P). Initialize $UA \leftarrow \emptyset$, $PA \leftarrow \emptyset$ Then, proceed to Step 2.
2. Take next item (t', P') from the buffer:
   - Among the permissions P', identify those permissions common for all team members t'. We say they are common if at least the percentage of team members having permission is larger than a given θ, where θ is set :
     $$\Pi(t') \leftarrow \{p \in P' : \omega(t', p) \geq \theta\}$$
     where
     $$\omega(t', p) = \frac{|u : u \in U, t' \in M(u), (u, p) \in UP|}{|u : u \in U, t' \in M(u)|}$$
     is ratio of users in t' having permission p.
   - Define a new role r' for these permissions $\Pi(t')$, i.e.
     $$PA \leftarrow PA \cup \{(r', p)\}_{p \in \Pi(t')}$$
     The role is attributed by the membership of that team.
   - Add all child teams to the buffer. Their related permissions are the related permissions of the current team without the permissions in the given role. Formally, for all $t'' : (t', t'') \in H$ we add to buffer $(t'', P' \backslash \Pi(t'))$
   - If the buffer is empty, stop. If not, iterate step 2.

To illustrate the algorithm, consider a simple hierarchy of teams $T = \{t_0, t_1, t_2\}$ where t_0 is parent to t_1 and t_2. Team t_1 has 5 users, t_2 has 5 users, and t_0 has two additional users (manager and secretary), who do not belong to either t_1 or t_2. Assume that the permissions p_0, p_1, p_2, p_3 are assigned as shown in Table 2. Assume that $\theta = 0.8$, then the roles and attributions will be the following: the role for t_0 will contain p_0

and p_1, and the role will be inherited by the child teams as well, t_1 will get p_2, and t_2 will get p_3. The assignment follows the Principle of Least Privilege (PoLP), i.e., ensures that access rights are assigned only to those, who needs them.

Table 2: Example of UP.

U	$M(u)$	p_0	p_1	p_2	p_3
u_0	t_0	1	1	0	0
u_1	t_0	1	1	0	0
u_2	(t_0,t_1)	1	1	1	0
u_3	(t_0,t_1)	1	1	1	0
u_4	(t_0,t_1)	1	1	1	0
u_5	(t_0,t_1)	1	1	1	0
u_6	(t_0,t_1)	1	1	1	0
u_7	(t_0,t_2)	1	1	0	1
u_8	(t_0,t_2)	1	1	0	1
u_9	(t_0,t_2)	1	1	0	1
u_{10}	(t_0,t_2)	1	1	0	1
u_{11}	(t_0,t_2)	1	1	0	0

5 CASE STUDY

We worked with data from a large financial institution, serving approximately 15,000 users, including agents. The current access control model is a mix of different Access control models, mainly Identity-based access control and Attribute Based Access Control. Currently, all data about users and their permissions are held in a control database, and they are then provisioned to different applications. Sometimes the permissions are set by the application administrators; sometimes, they are assigned to users automatically based on their attributes. Currently, permission management is carried out per application; no permission groups would contain permissions from various applications. The current model is not effective nor salable for the big company. As a result, the process of receiving necessary permissions for the new employees or in case of changing roles often takes weeks and is very ineffective.

The company's ultimate objective was transitioning to a Role-Based Access Control (RBAC) model. To achieve this, we have been tasked with developing roles that can be automatically assigned based on user attributes.

5.1 Role-Mining in Practise

Our objective was to mine roles only for back-office employees, thus the $\overline{UP}$ sparse matrix was of shape 3827*4882. Firstly, we tried the ML approach. To reduce dimensionality, we used 350 SVD components, which allowed us to preserve 91% of total variability in data.

In our k-means clustering experiments, we tested with $k = 350$ and $k = 450$, resulting in 66 and 61 roles, respectively. Despite the high number of clusters, many were excluded in the preparation stage for logistic regression. As previously discussed (Subsection 4.2.2), roles with fewer than m users were removed since training a meaningful logistic model with too few observations is not feasible. We chose $m = 10$. Therefore, when k was set to less than 350, the algorithm produced a very small number of roles, leading to a lower overall WSC.

In order to binarize $\overline{Q}$ to $\overline{PA}$ the optimization library Hyperopt (Bergstra et al., 2013) was used to optimize the threshold η based on WSC minimisation. To construct the Y for machine learning, it's important to choose the right λ value. Setting $\lambda = 1$ creates a lot of precise and strict roles, eliminating the need for manual checks by line managers or security experts. On the other hand, a lower λ value introduces more flexibility and fewer roles, improving generalization but also increasing the risk of NUPA or false positives, which require manual review. After trying various λ values and consulting with the business, we decided on $\lambda = 0.9$ to make the algorithm more flexible. The same reasoning was used for choosing the θ value for the HTPA algorithm. Another strategy to determine the appropriate values for θ and λ would be including them into the Hyperopt optimization based on WSC, but it is beyond the scope of this article.

For the logistic-regression-based attribution, we used these features about the user:

1. CDUUserTypeID - type of user (employee, technical account, test account, external, etc.)
2. QualificationCodeID - qualification/certification
3. ProfessionID - user profession code
4. GroupOfEmployeesID - user employment type - full-time, part-time etc.
5. OrganizationUnitID - the user's organizational unit

To test the logit model performance we splited a data set to train (70%) and test (30%). The WSC criterion was afterwards calculated on the test data.

After that we tried the second approach for the role-mining - HTPA. The algorithm was programmed in Python utilizing the pandas multiindex functionality. As a result we set paths in H to Pandas multiindex and looped through it. The threshold percentage for defining common permissions θ was set to 0.9 after experimenting with different values and consulting with the business. Finally, the WSC was calculated for the test set data.

5.2 Results

In Table 3 the Weighted Structural Complexity (WSC) for the both solution approaches can be found as well as the secondary metric Covering rate (CR). According to the literature (Molloy et al., 2008), the WSC weights were set as $W = [1,1,1,1,1,1]$. Since we do not introduce any additional hierarchy of roles RH, $|t_{reduce}(RH)| = 0$ for both algorithms.

We can clearly see that our Hierarchical Team Permission Analysis based on the companies' hierarchy clearly outperforms the roles created and attributed using ML approach according to the WSC criterion. The improvement is drastic: -49 % WSC for Clustering with $k = 350$ and -47 % WSC for Clustering with $k = 450$. The biggest advantage of Hierarchical Team Permission Analysis is an extensive decline in costly $|NUPA|$ compared to ML approach, particularly: -95 % and -94 % (!) decline compared to clustering models. There is also a significant decrease in number of user assignments $|UA|$: -42 % and -46 %, and a small decrease in $|DUPA|$: -8 %.

On the other hand, there is notable increase in the number of roles compared to ML roles. Due to setting the minimal number of users for a ML role $m = 10$, the majority of roles were removed. The number of permission assignments $|PA|$ is similar to ML results with $k = 350$, and 16 % higher compared to $k = 450$ suggesting, that HTPA can bring somewhat higher overlap between roles.

Considering the secondary metric - the Covering rate, which tells us, how many original permissions were covered by roles, the HTPA slightly outperformed the ML methods: +6 %.

Another benefit of using HTPA, based on companies' hierarchy for the role-mining, is that it provides meaningful roles with clear business interpretation: we receive specific roles for different departments and teams. This improves the probability, that the defined roles will be actually used in practise by the security administrators.

A further advantage of our algorithm is, that it provides very clear assignment rules for the created roles: when new employee arrives to the team, he/she is going to automatically receive the roles according to the hierarchy of his team with minimal errors and security risks. On the other hand, Machine Learning approach does not allow for such clear, simple and almost error-less attribution due to its probabilistic nature, and classical role-mining algorithms do not allow for any automatic attribution. Moreover our algorithm is quite simple and thus trustworthy for the business users.

Table 3: WSC for ML approach and HTPA.

	Clusters (k=350)	HTPA vs Clust. (k=350)	Clusters (k=450)	HTPA vs Clust. (k=450)	HTPA
$\lvert R\rvert$	66	479%	61	526 %	382
$\lvert UA\rvert$	8,579	-42 %	9,275	-46 %	4,989
$\lvert PA\rvert$	4,944	4 %	4,447	16 %	5,147
$\lvert t_{reduce}(RH)\rvert$	0	0 %	0	0 %	0
$\lvert DUPA\rvert$	37,198	-8 %	37,108	-8 %	34,189
$\lvert NUPA\rvert$	40,586	-95 %	37,303	-94 %	2,084
$wsc(\gamma, W)$	91,373	-49 %	88,194	-47 %	46,791
CR	58 %	6 %	58 %	6 %	62 %

6 CONCLUSION

Traditional RBAC role-mining algorithms output a mathematically optimized set of roles, nevertheless, they often lack a reasonable interpretation, neither allow for assigning users to roles. ML models in access control could be risky due to their probabilistic nature and some level of inaccuracy. Due to such drawbacks, the above-mentioned methods are not well suited for our research question, which was finding the RBAC roles as well as a mechanism of attributing these roles to users efficiently and precisely.

In this paper, we introduce two possible solutions for solving the business requirement. Fist approach was using Machine Learning methods, particularly: 1) reduce dimensionality of data; 2) apply k-means clustering, where centers are roles; 3) predict/attribute roles with a logistic regression for every role. The second approach was creating a Hierarchical Team Permission Analysis algorithm, which was based on the company's hierarchical structure.

The results of real-life application of these methods show, that the HTPA algorithm produce roles with higher Covering rate, and fewer corrective unassignments and assignments. The attribution of roles based on companies' hierarchy is straightforward and error-resistant. Another benefit of the HTPA algorithm is, that it's quite simple, and provides roles with clear interpretation, thus making them trustworthy for the business users.

Nevertheless, there are still some ideas to explore, for example, to combine both suggested solutions, particularly applying the ML approach to the data, which were not covered by the HTPA roles, and thus improve the covering rate. Another future research idea is to fine-tune the threshold for assigning users to roles, λ, and the threshold for identifying shared permissions in HTPA, θ, aiming to reduce WSC. Additionally, achieving a better balance between two kinds of errors, specifically DUPA (false negatives) and NUPA (false positives), could be investigated further by using varied weights during the optimization process.

REFERENCES

Abolfathi, M., Raghebi, Z., Jafarian, J., and Banaei-Kashani, F. (2021). A scalable role mining approach for large organizations. pages 45–54.

Al-Kahtani, M. and Sandhu, R. (2004). Rule-based rbac with negative authorization. In *20th Annual Computer Security Applications Conference*, pages 405–415.

Alayda, S., Almowaysher, N., Humayun, M., and Jhanjhi, N. (2020). A novel hybrid approach for access control in cloud computing. *International Journal of Engineering Research and Technology*, 13:3404–3414.

Assent, I. (2012). Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):340–350.

Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA. PMLR.

Cavoukian, A., Chibba, M., Williamson, G., and Ferguson, A. (2015). The importance of abac: attribute-based access control to big data: privacy and context. *The Privacy and Big Data Institute, Canada*.

Di Pietro, R., Colantonio, A., and Ocello, A. (2012). *Role mining in business: taming role-based access control administration*. World Scientific.

Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., and Tarjan, R. (2008). Fast exact and heuristic methods for role minimization problems. pages 1–10.

Jayasundara, S., Arachchilage, N., and Russello, G. (2024). Vision: ”accessformer”: Feedback-driven access control policy generation framework.

Karp, A., Haury, H., and Davis, M. (2010). From abac to zbac: The evolution of access control models. *ISSA (Information Systems Security Association). Journal*, 8:22–30.

Kern, A., Kuhlmann, M., Schaad, A., and Moffett, J. (2002). Observations on the role life-cycle in the context of enterprise security management. pages 43–51.

Köppen, M. (2000). The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8.

Kuhn, D., Coyne, E., and Weil, T. (2010). Adding attributes to role-based access control. *Computer*, 43:79–81.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., and Lobo, J. (2008). Mining roles with semantic meanings. pages 21–30.

Ni, Q., Lobo, J., Calo, S., Rohatgi, P., and Bertino, E. (2009). Automating role-based provisioning by learning from examples. pages 75–84.

Nobi, M., Gupta, M., Praharaj, L., Abdelsalam, M., Krishnan, R., and Sandhu, R. (2022a). Machine learning in access control: A taxonomy and survey.

Nobi, M. N., Krishnan, R., Huang, Y., Shakarami, M., and Sandhu, R. (2022b). Toward Deep Learning Based Access Control. *CODASPY ’22: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, page 143–154.

Olabanji, S., Olaniyi, O., Adigwe, C., Okunleye, O., and Oladoyinbo, T. (2024). Ai for identity and access management (iam) in the cloud: Exploring the potential of artificial intelligence to improve user authentication, authorization, and access control within cloud-based systems. *Asian Journal of Research in Computer Science*, 17:38–56.

Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Schlegelmilch, J. and Steffens, U. (2005). Role mining with orca. pages 168–176.

Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Vaidya, J., Atluri, V., and Warner, J. (2006). Roleminer: mining roles using subset enumeration. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 144–153.

Zhang, D., Ramamohanarao, K., and Ebringer, T. (2007). Role engineering using graph optimisation. pages 139–144.

Zhou, L., Su, C., Li, Z., Liu, Z., and Hancke, G. P. (2019). Automatic fine-grained access control in scada by machine learning. *Future Generation Computer Systems*, 93:548–559.