Titre: Title:	Role mining sensible à l'accumulation de privilèges
Auteur: Author:	Vincent Bittard
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bittard, V. (2025). Role mining sensible à l'accumulation de privilèges [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/70214/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Role mining sensible à l'accumulation de privilèges

VINCENT BITTARD
Département de Génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Novembre 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Role mining sensible à l'accumulation de privilèges

présenté par **Vincent BITTARD**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Alejandro QUINTERO, président
Nora BOULAHIA-CUPPENS, membre et directrice de recherche
Frédéric CUPPENS, membre et codirecteur de recherche
Ranwa AL MALLAH, membre

DÉDICACE

Je remercie mes collègues du laboratoire, Vi, Basile, Amine, Florian, Erwan, Maxime, pour leur bienveillance et leur présence tout au long de ce parcours.

Je remercie profondément tous mes amis qui, chacun à leur manière, m'ont soutenu, encouragé et apporté de précieux instants de réconfort. Un merci tout particulier à Auriane et Dorian, avec qui j'ai partagé cette aventure en maitrise au quotidien pendant deux années, pour tous ces moments passés ensemble et leur aide inestimable.

Je remercie grandement ma famille et Julien pour leur soutien indéfectible, leur présence attentive et leurs encouragements constants, qui m'ont permis de traverser les périodes de doute, de surmonter les moments les plus éprouvants et de poursuivre ce travail avec confiance.

Enfin, je dédie ce mémoire de recherche à Sébastien Foucher.

REMERCIEMENTS

Je remercie mes directeurs de recherche Frédéric et Nora Cuppens pour m'avoir donné l'opportunité de développer ce sujet de maitrise au sein du projet sur la menace interne.

Je remercie mes encadrants de recherche Jean-Yves Ouattara, Sara Imene Boucetta, et plus particulièrement Rim Ben Salem et Ahmed Bouzid pour m'avoir appuyé dans mon travail de rédaction à maintes reprises.

Je remercie mes superviseurs industriels Adel Benlagra, François Charest et en particulier Mouna Selmi, qui a encadré pendant plus d'un an l'implémentation de mon projet de recherche et fourni une aide précieuse tout au long de mon stage.

Je tiens à exprimer ma gratitude envers MITACS, Banque Nationale, Desjardins, Mondata et Qohash pour leur soutien et leur contribution inestimables à cette recherche.

Je remercie Sébastien Foucher pour ces travaux préalables qui m'ont aidé dans ma recherche.

Je remercie enfin le jury pour l'attention portée à l'évaluation de mon travail.

RÉSUMÉ

Le Role Mining (RM) extrait des structures de contrôle d'accès fondées sur des rôles (RBAC) à partir des attributions de permissions aux utilisateurs afin de réduire la charge administrative. Cependant, les approches existantes partent généralement du principe que les ensembles de données utilisées pour miner des rôles sont propres, alors que les systèmes réels souffrent d'anomalies telles que l'accumulation de privilèges, aussi appelé privilege creep.

L'approche proposée vise à détecter les utilisateurs susceptibles d'être concernés par l'accumulation de privilèges et qui doivent être examinés en priorité, ainsi qu'à identifier les attributions de permissions légitimes à exprimer en RBAC, réduisant ainsi la complexité de la gestion. Cette approche consiste en une procédure à deux étapes : nettoyer la matrice d'assignation des permissions utilisateur (UPA) à l'aide de clustering et d'une analyse statistique, puis construire un état RBAC à l'aide d'un algorithme de role mining classique.

L'approche proposée permet d'obtenir une précision moyenne de 90 % dans la détection de l'accumulation des privilèges et une correction de plus de 95 % de l'accumulation des privilèges, évaluée sur des jeux de données synthétiques. L'évaluation sur des jeux de données réels montre une réduction moyenne d'un facteur 4 des rôles requis tout en conservant une couverture de la matrice User-Permission Assignment matrix (UPA) d'au moins 80 %.

ABSTRACT

Role Mining (RM) extracts Role-Based Access Control (RBAC) structures from user-permission assignments to reduce administrative overhead. However, existing approaches generally assume that the datasets used for mining roles are clean, while real systems suffer from anomalies like privilege creep (PC), the gradual accumulation of outdated permissions, permissions that should have been revoked.

My approach aims to detect users affected by privilege creep for prioritized access review, while identifying legitimate permission assignments that can be expressed in RBAC, thereby reducing management complexity. This approach consists of a two-step procedure: cleaning the User-Permission Assignment (UPA) matrix using clustering and statistical analysis, then constructing an RBAC state using a conventional role mining algorithm.

The proposed approach achieves an average accuracy of 90% in detecting privilege creep and is able to correct over 95% of privilege creep instances, as evaluated on synthetic datasets. The evaluation on real-world datasets shows an average reduction by a factor of 4 in the required roles while maintaining a UPA matrix coverage of at least 80%. This reduction in role count is comparable to or exceeds the performance of established methods such as delta RMP for the datasets evaluated.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Abréviations standard et acronymes issus de la littérature

ABAC	Attribute-Based Access Control
ACL	Access Control List
AHC	Agglomerative Hierarchical Clustering
AWS	Amazon Web Services
BNMF	Binary Non-Negative Matrix Factorization
CISA	Cybersecurity & Infrastructure Security Agency
CM	CompleteMiner
CSP	Complexité Structurelle Pondérée
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDM	Disjoint Decomposition Model
δ-RMP	δ Role Mining Problem
DLBAC	Deep Learning Based Access Control
ERBAC	Enterprise Role Based Access Control
FM	FastMiner
GIA	Gestion des Identités et des Accès
GO	Graph-based Optimization
GNN	Graphical Neural Network (réseaux de neurones graphiques)
HM	HierarchicalMiner
IAM	Identity and Access Management
IEEE	Institute of Electrical and Electronics Engineers
IF	Isolation Forest
LOF	Local Outlier Factor
MAC	Multi-Assignment Clustering
MinNoise-RMP	Minimal Noise Role Mining Problem
NDCT	Normalized Difference Curve Technique
NMF	Non-Negative Matrix Factorization
NIST	National Institute of Standards and Technology
OCC	Office of the Comptroller of the Currency
ORCA	Ontology-based Role-mining framework with Clustering Analysis
PC	Privilege Creep
PCA	Principal Component Analysis

PFA	Principal Feature Analysis
RBAC	Role Based Access Control
RM	Role Mining
RMP	Role Mining Problem
SVD	Singular Value Decomposition
TSVD	Truncated Singular Value Decomposition
UPA	User-Permission Assignment matrix
WSC	Weighted Structural Complexity

Acronymes définis par l'auteur

RPL	Rétention des Permissions Légitimes
EP	Expression des Permissions
CPC	Correction du Privilege Creep
PDPC	Précision de la Détection du Privilege Creep
ENR	Écart du Nombre de Rôles

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte de menace interne

Les menaces internes désignent des actes malveillants commis par des personnes qui possèdent des autorisations d'accès légitimes au sein d'une organisation. Ces actes peuvent avoir des conséquences graves sur les ressources numériques et physiques de l'entreprise.

Dans un livre de Stolfo et al [2], un acteur de menace interne désigne tout individu disposant d'autorisations légitimes pour accéder aux systèmes et ressources numériques de l'organisation. Cette catégorie comprend toute personne habilitée à consulter, modifier ou gérer les configurations informatiques, les données ou les applications de l'entreprise, privilèges qui ne sont pas accordés au public externe. Cette définition couvre non seulement le personnel permanent de l'organisation, mais également les employés temporaires, les bénévoles et les prestataires externes, la portée exacte dépendant du domaine d'activité spécifique de l'entreprise concernée.

On peut relier cette notion de personne malveillante interne à l'entreprise sous le terme d'initié, "insider" en anglais, lorsque l'acte malveillant est intentionnel. Ces initiés peuvent avoir plusieurs motifs comme le souligne Fortinet [3] : de nombreuses menaces internes intentionnelles sont motivées par un désir de vengeance envers l'entreprise suite à un sentiment d'injustice ou d'attentes insatisfaites, notamment l'absence de prime ou de promotion espérée. Cybersecurity & Infrastructure Security Agency (CISA) [4] propose une perspective complémentaire en identifiant les motivations d'initiés qui cherchent à porter préjudice à leur organisation dans un objectif de gain personnel ou en réaction à un grief particulier. L'agence observe notamment que plusieurs initiés sont poussés à la vengeance par un sentiment de manque de reconnaissance perçu, qu'il s'agisse de promotions refusées, de primes non accordées, d'opportunités de déplacement manquées ou encore de licenciements.

Cependant, cette définition n'inclut pas les acteurs de la menace interne qui agissent de façon non intentionnelle. CISA [4] donne deux catégories de ces menaces internes :

- Négligence, un acteur de ce type expose une organisation à une menace par manque de vigilance. Les employés négligents connaissent généralement les politiques de sécurité et/ou informatiques, mais choisissent de les ignorer, créant ainsi un risque pour l'organisation. Les exemples incluent le fait d'égarer ou perdre un dispositif de stockage portable contenant des informations sensibles ou ignorer les messages demandant d'installer de nouvelles mises à jour et correctifs de sécurité.

- Accidentel, un acteur de ce type cause par erreur un risque non intentionnel pour une organisation. Les exemples incluent une erreur de frappe dans une adresse courriel et l'envoi accidentel d'un document commercial sensible à un concurrent, cliquer sans le savoir ou par inadvertance sur un hyperlien, ouvrir une pièce jointe dans un courriel d'hameçonnage contenant un virus, ou éliminer de façon inappropriée des documents sensibles.

Ainsi donc, un autre acteur intentionnellement malveillant tire profit des acteurs internes dont les agissements non intentionnels mettent en péril la sécurité de l'entreprise.

1.2 Risques et impacts

Les incidents liés aux menaces internes, qu'ils soient intentionnels ou non, peuvent engendrer des dommages considérables. Ces préjudices incluent le vol, la divulgation et la détérioration de données sensibles, la désactivation malveillante des services critiques, la saturation des réseaux ou des serveurs, l'interruption des processus métier essentiels, ou encore l'assistance apportée aux attaquants externes par la création de points d'accès non autorisés.

Le coût total moyen des incidents de menace interne est passé de 8,3 millions de dollars US en 2018 à 16,2 millions de dollars US en 2023 selon le rapport global 2023 du Ponemon Institute sur le coût des menaces internes [5], soit un doublement en seulement cinq ans.

En 2024, la situation s'est fortement dégradée par rapport à 2023 : Un rapport fondé sur sondage de 467 professionnels de cybersécurité par Cybersecurity Insiders montre que 17% des organisations ont déclaré n'avoir subi aucune attaque interne : incidents d'exposition, de perte, de fuite et de vol de données causés par des acteurs malveillants [6]. C'est une diminution significative par rapport aux 40% sondés en 2023.

On peut citer plusieurs cas célèbres pour se rendre compte de la gravité de certains accidents qui ternissent l'image d'une entreprise en plus de faire peser de lourdes pénalités financières.

Le premier exemple est celui de la fuite de données de Capital One, perpétrée par une ancienne employée de Amazon Web Services (AWS) qui a profité d'un pare-feu mal configuré, utilisé par Capital One pour protéger son déploiement AWS. Ce pare-feu s'était vu accorder des permissions excessives sur l'instance AWS (la capacité de lire tous les fichiers stockés) et était vulnérable. L'attaquante a exploité la vulnérabilité en question pour dérober les données de 100 millions de clients américains et 6 millions de clients canadiens de Capital One [7]. Suite à cet incident, l'entreprise a dû payer une amende de 80 millions de dollars US à l'Office of the Comptroller of the Currency (OCC) [8] ainsi que 190 millions de dollars US suite à un recours collectif [9] faisant s'élever la note totale à 270 millions de dollars US. C'est sans

compter la perte de confiance des clients de Capital One qui a probablement eu un impact financier impossible à calculer précisément.

Un cas plus récent qui n'a pas encore été jugé est celui de la fuite de données de Tesla ayant eu lieu en 2023. Deux anciens employés de Tesla agissant en lanceurs d'alerte ont fait fuiter plus de 23 000 documents internes à un média allemand [10] [11]. Les données confidentielles incluaient les informations d'identification personnelle de plus de 75 000 employés, des secrets de production et des informations financières des clients pour un total de 100 Giga-octets de données. Les amendes potentielles en vertu du RGPD auraient pu atteindre des milliards de dollars. L'entreprise fait également face à un recours collectif des employés toujours en cours à la date d'écriture de ce mémoire [12].

Dans tous ces exemples, les acteurs malveillants ont utilisé des privilèges auxquels ils avaient accès au sein de leur entreprise. Dans certains cas, c'est une conséquence directe d'un phénomène souvent observé en contrôle d'accès : au fil du temps, la qualité des systèmes de Gestion des Identités et des Accès (GIA) se dégrade, l'accumulation de privilèges représentant une anomalie clé dans le contexte de menace interne : une accumulation progressive d'autorisations résultant de transitions professionnelles, de changements organisationnels, d'affectations temporaires, etc. Si ce problème peut être résolu manuellement dans les petites organisations, sa complexité augmente considérablement dans les grandes entreprises, d'autant plus que la revue des accès est souvent fragmentée et déléguée.

1.3 Solutions

Une étude récente par Marquis [13] énonce que les menaces internes constituent un défi majeur pour la sécurité des bases de données organisationnelles, nécessitant des mesures de protection robustes et adaptatives. Le contrôle d'accès basé sur les rôles Role Based Access Control (RBAC) représente une solution prometteuse pour faire face à ces risques en limitant l'accès aux données selon la fonction de chaque utilisateur au sein de l'organisation. Cette recherche examine l'efficacité des systèmes RBAC pour réduire les menaces internes dans différents secteurs d'activité, notamment la technologie, la finance, la santé et les organismes gouvernementaux, en s'appuyant sur une approche quantitative utilisant une enquête auprès de professionnels responsables de la sécurité des bases de données.

Les résultats révèlent que le RBAC contribue effectivement à diminuer les accès non autorisés et les violations de données, réduisant ainsi considérablement les menaces internes. Cependant, la mise en œuvre présente certains défis, notamment la complexité de la définition des rôles et l'adaptation aux besoins d'accès évolutifs. L'étude souligne que l'efficacité du RBAC dépend

de son amélioration continue, de l'intégration de technologies avancées comme l'apprentissage automatique, et de son adaptation aux contextes organisationnels spécifiques, recommandant ainsi des programmes d'amélioration continue et une formation spécialisée pour optimiser ces systèmes de sécurité.

Les administrateurs système structurent aussi les permissions données aux utilisateurs pour réduire les coûts liés à la GIA. Ils utilisent alors le modèle RBAC ou Attribute-Based Access Control (ABAC), en regroupant les utilisateurs ayant des besoins similaires en matière d'accès à l'information. Cela permet une configuration du contrôle d'accès plus extensible et plus facile à maintenir. Le processus qui sert à construire cette structure s'appelle "role mining" puisqu'on essaie d'inférer les rôles des utilisateurs en fonction de leurs permissions, attributs, etc. Cependant, les approches récentes en matière de role mining se sont concentrées sur des méthodes conçues pour des jeux de données propres, sans anomalies. Dans les systèmes de contrôle d'accès réels, cette hypothèse est rarement vérifiée. Ceci crée un problème majeur : les permissions anormales sont répliquées dans la structure RBAC et rendent donc le contrôle d'accès inutile lorsque celui-ci ne correspond pas aux politiques de sécurité mises en place.

1.4 Objectifs de recherche

Mettre au point une méthode de role mining capable de détecter et de corriger les anomalies principales rencontrées en GIA tout en proposant des recommandations pour former des rôles et structurer la gestion des accès.

Développer un algorithme de génération de données synthétiques paramétrable qui prend en compte l'ajout d'anomalies comme le bruit et l'accumulation de privilèges.

Évaluer les performances de la méthode de role mining avec détection de l'accumulation de privilèges mise au point sur des jeux de données synthétiques générés, et sur des jeux de données réels en combinant des métriques nouvellement développées avec des métriques établies.

1.5 Plan du mémoire

Le deuxième chapitre présente une revue de la littérature des approches explorées en role mining et les techniques utilisées pour évaluer ces approches. Le troisième chapitre décrit la méthode de génération de jeux de données synthétiques proposée. Le quatrième chapitre explique la méthode de role mining proposée, capable de détecter et corriger les instances d'accumulation de privilèges. Le cinquième chapitre est dédié à l'évaluation sur jeux de données réels et synthétiques de la méthode de role mining proposée. Enfin le sixième chapitre constitue la conclusion des travaux, incluant la limites, les pistes d'améliorations et les travaux futurs.

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 Role Mining

2.1.1 Définition formelle

Le National Institute of Standards and Technology (NIST) [14] apporte une définition formelle au contexte commun du role mining :

- Soit U, P l'ensemble des utilisateurs et l'ensemble des permissions respectivement.
- Soit $UPA \subseteq U \times P$ la matrice binaire d'assignation utilisateur-permission, pouvant être vue comme une fonction multivaluée [15] de U dans P. Cette matrice est, dans la plupart des cas, vue comme une matrice binaire où un 1 indique une assignation de permission à un utilisateur et un 0 indique l'absence d'assignation.

Le NIST formalise aussi le basic Role Mining Problem (RMP) en un problème de décomposition matricielle binaire :

Étant donné le contexte commun du role mining, trouver un ensemble de rôles R et deux matrices binaires $UA \subseteq U \times R$, la matrice d'affectation utilisateur-rôle et $PA \subseteq P \times R$, la matrice d'affectation rôle-permission, où $UPA = UA \otimes PA$ en minimisant $|R|$.

L'opération $\otimes$ n'est pas une multiplication matricielle au sens algébrique classique (produit scalaire ou produit matriciel standard), mais une opération booléenne d'affectation, souvent appelée produit booléen ou composition. On le calcule comme suit :

$$UPA[u, p] = \bigvee_{r \in R} \Big(UA[u, r] \wedge PA[r, p] \Big) \tag{2.1}$$

Avec $\bigvee$ le OU logique sur un ensemble, $\wedge$ le ET logique.

Par abus de langage, on désigne les assignations utilisateur-permission par le terme assignation. De même, on désigne les affectations utilisateur-rôle et rôle-permission par le terme affectation.

2.1.2 Autres formulations et algorithmes

D'autres formulations du RMP existent et servent à gagner en pertinence de solution, en temps de calcul, etc.

Zhang et al. [16] proposent une formulation du problème par graphe biparti [17] où les utilisateurs, les rôles et les permissions sont représentés comme des nœuds, et les assignations entre eux sont représentées comme des arêtes. Le role mining devient alors un problème d'expression de graphe biparti en graphe triparti. L'article explore une approche hybride où les rôles sont partiellement définis au niveau de l'administration. L'objectif principal est de minimiser le coût d'administration après que l'état RBAC a été construit. Cela se traduit par un objectif de nombre minimal de rôles et d'arêtes dans le graphe biparti. Cette formulation est particulièrement intéressante, car elle permet d'utiliser des techniques d'optimisation de graphe. La figure 2.1 donne un exemple de cette approche.

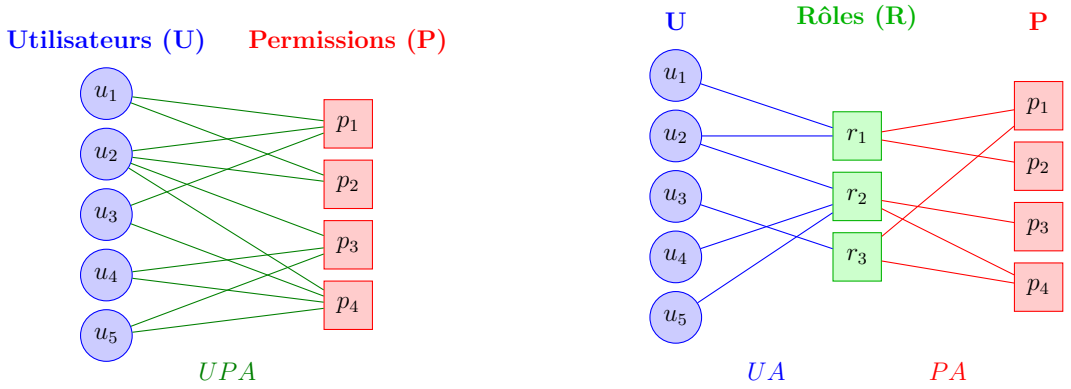


Figure 2.1 Comparaison de l'expression d'une matrice UPA sous forme de graphe biparti 2.1a, et forme de graphe triparti 2.1b.

Le problème s'énonce alors :
Étant donné le contexte commun du role mining, trouver un ensemble de rôles R et deux matrices binaires $UA \subseteq U \times R$ et $PA \subseteq P \times R$ où $UPA = UA \otimes PA$ en minimisant $|R| + \|UA\|_1 + \|PA\|_1$.

Cette approche est améliorée par Vaidya et al. [18] en formalisant et en nommant le problème edge-RMP. Ils montrent que ce problème est NP-complet et proposent une solution algorithmique pour le résoudre. Il s'agit toujours d'un problème visant à minimiser le nombre total d'arêtes dans un graphe biparti, mais sans la contrainte sur les rôles. Le problème s'énonce alors de façon similaire :
Étant donné le contexte commun du role mining, trouver un ensemble de rôles R et deux matrices binaires $UA \subseteq U \times R$ et $PA \subseteq P \times R$, où $UPA = UA \otimes PA$ en minimisant $\|UA\|_1 + \|PA\|_1$.

Diverses approches algorithmiques sont utilisées pour trouver une solution à ces problèmes de role mining, voici les plus connues encore utilisées à ce jour regroupées en trois catégories :

- La décomposition de matrice, visant à factoriser directement la matrice UPA. Au sein de cette catégorie, CompleteMiner (CM) [19] offre une approche exhaustive en explorant toutes les intersections des ensembles de permissions portées par les utilisateurs possibles pour garantir l'identification de tous les rôles potentiels, au prix d'un coût de calcul et de mémoire très élevé. Pour pallier ce problème, FastMiner (FM) [19] a été proposé comme une alternative heuristique plus rapide : limiter la recherche de rôles à un sous-ensemble d'intersections, ne prenant en compte que les intersections impliquant au plus 2 configurations utilisateur, et se concentrer sur les utilisateurs ayant des permissions similaires. L'algorithme gagne en efficacité pour les grands systèmes, passant d'une complexité exponentielle à polynomiale, bien que cette optimisation empêche parfois de découvrir certains rôles optimaux.

- Basée sur le clustering, qui regroupe les utilisateurs ou les permissions selon une métrique de similarité. HierarchicalMiner (HM) [20] se distingue en utilisant des techniques de clustering hiérarchique pour identifier des rôles, mais aussi les organiser dans une structure arborescente. Cette hiérarchie modélise plus fidèlement les relations entre les fonctions au sein d'une organisation, bien que sa pertinence dépende fortement de la mesure de similarité choisie. Pour enrichir davantage la signification des rôles découverts, des approches comme Ontology-based Role-mining framework with Clustering Analysis (ORCA) [21] combinent le clustering avec des ontologies, intégrant ainsi le contexte organisationnel pour générer des rôles sémantiquement plus riches.

- Basée sur les graphes, offrant une modélisation flexible du problème. Des algorithmes comme HPr [1] ou Graph-based Optimization (GO) [16] cherchent à identifier des "communautés denses" d'utilisateurs, qui correspondent à des rôles potentiels. Cette méthode transforme le role mining en un problème d'optimisation, permettant d'intégrer diverses contraintes spécifiques au système.

Toutes ces approches visent une correspondance exacte entre la matrice UPA originelle et la matrice reconstruite à partir des matrices UA et PA. Cependant, comme Vaidya et al. [22] l'ont noté, cette méthode peut produire un nombre trop important de rôles, rendant l'avantage administratif apporté par la structure RBAC moindre. C'est pourquoi la version généralisée du RMP autorise des déviations.

2.2 Noise Role Mining

On entend par déviation que la matrice UPA n'a pas à être exprimée entièrement, voire qu'on peut rajouter des assignations préalablement non existantes. Cette approche permet de pallier un premier type d'anomalie rencontré dans les systèmes de GIA : le bruit. Cette notion de bruit est ce qui donne à la version généralisée du problème le nom de "Noise Role Mining" ou role mining en présence de bruit.

2.2.1 Définition du bruit

Vaidya et al. [23] définissent le bruit observable sur la matrice UPA comme des permissions qui ne sont pas enregistrées dans le système de GIA comme elles devraient l'être. C'est-à-dire, soit une autorisation étant enregistrée comme un refus ou un refus étant enregistré comme une autorisation. On appelle ces anomalies bruit général : des inversions de bits sur la matrice UPA, essentiellement une permission incorrectement révoquée ou accordée par un administrateur de sécurité. Ils classifient ensuite ce bruit en deux catégories :

- Bruit additif, qui désigne les permissions incorrectement ajoutées dans la matrice UPA qui n'ont pas été supprimées. Habituellement, ces permissions sont accordées pour une tâche temporaire et ont été oubliées.
- Bruit soustractif, qui désigne les permissions incorrectement supprimées de la matrice UPA qui devraient y être ajoutées. Habituellement, ces permissions manquantes proviennent du processus d'approvisionnement (access provisioning) lorsque quelqu'un n'obtient pas les permissions nécessaires pour accomplir ses tâches.

Il est clair que le bruit soustractif est habituellement moins fréquent que le bruit additif en raison du problème de disponibilité qu'il cause. Molloy et al. [24] proposent une autre classification comme suit :

- Bruit de correction : les assignations qui impactent la sécurité d'un système. Essentiellement, ce bruit est défini comme les erreurs de type I (faux positifs) et de type II (faux négatifs) qui correspondent respectivement au bruit additif et soustractif de Vaidya et al. [23]
- Bruit d'applicabilité RBAC : les assignations que l'administrateur souhaite conserver comme des exceptions, et qui ne devraient pas être exprimées dans le système RBAC. Ceci découle du besoin de flexibilité dans le contrôle d'accès, permettant à

l'administrateur d'adapter les permissions dynamiquement en fonction du contexte ou des exigences temporaires. Ce ne sont pas à proprement parler des erreurs, mais elles peuvent être indiscernables du bruit de correction sans contexte.

La définition de Molloy et al. [24] est donc plus large que celle de Vaidya et al. [23].

Il est important de noter que la quantité de bruit rajoutée sur un jeu de données quelconque, traduit sous forme matricielle UPA, est généralement exprimée en pourcentage du nombre d'assignations totales. Autrement dit, le nombre d'assignations "bruités" à rajouter sur le jeu de données en fonction du pourcentage de bruit p est $p \times |UPA|$. Par abus de langage, ce pourcentage s'appelle fréquemment niveau de bruit. Les niveaux de bruit usuels introduits sont de l'ordre de 1% à 15% pour Molloy et al. et jusqu'à 30% pour Vaidya et al.

2.2.2 Premières solutions

Vaidya et al. [23] proposent deux approches pour atteindre cet objectif : le δ Role Mining Problem (δ-RMP) et le Minimal Noise Role Mining Problem (MinNoise). Ils permettent de miner des rôles directement sur des données dites "bruitées" en autorisant l'expression partielle de la matrice UPA, ce qui permet de tenir compte de potentielles anomalies. On appelle divergence la différence binaires d'assignation entre la matrice UPA d'origine et la matrice minée par l'algorithme. Le δ-RMP fixe le nombre maximal de divergences à δ, minimisant ainsi le nombre de rôles produits, tandis que le MinNoise fixe le nombre de rôles et minimise le nombre de divergences. Les deux approches proposées produisent moins de rôles que ce qui serait nécessaire pour exprimer pleinement les assignations, ce qui facilite la gestion de l'état RBAC résultant. Ainsi le problème de δ-RMP s'énonce :

Étant donné le contexte commun du role mining et un entier non nul δ, trouver un ensemble de rôles R et deux matrices binaires $UA \subseteq U \times R$ et $PA \subseteq P \times R$, où $UPA = UA \otimes PA$, tel que $||UPA - UA \times PA||_1 < \delta$ minimisant $|R|$.

Et le MinNoise-RMP :

Étant donné le contexte commun du role mining et un entier non nul k, trouver un ensemble d'exactement k rôles R et deux matrices binaires $UA \subseteq U \times R$ et $PA \subseteq P \times R$ où $UPA = UA \otimes PA$, en minimisant $||UPA - UA \times PA||_1$

Le Disjoint Decomposition Model (DDM) [25] [26] par Frank et al. utilise des contraintes d'expression afin d'orienter le role mining. En effet, avec cette approche les permissions sont assignées aux utilisateurs à travers des rôles "métier" et des rôles "fonctionnels", structure formalisée par Kern et al. [27]. Chaque utilisateur ne peut avoir qu'un seul rôle métier, et chaque permission ne peut appartenir qu'à une seule fonction, de facto qu'un seul rôle

fonctionnel. Les rôles fonctionnels peuvent en revanche être attribués à autant de rôles métiers que nécessaire pour exprimer les permissions. Cette disjonction du problème permet d'abstraire plus facilement l'attribution des rôles aux utilisateurs et permet aussi de corriger les erreurs d'assignation dans la matrice originelle en regroupant les utilisateurs similaires dans le même rôle métier. L'approche permet de corriger des niveaux de bruit uniformément aléatoires jusqu'à 10% des assignations totales sur la matrice UPA.

Le Multi-Assignment Clustering (MAC), d'abord introduit par Streich et Frank et al. [28] [29], exprime aussi partiellement la matrice UPA. Cette méthode s'inscrit dans la catégorie de role mining probabiliste. L'approche groupe les utilisateurs selon leurs permissions dans des clusters autorisés à se chevaucher. Des affectations probabilistes sont ensuite effectuées, en utilisant l'information sur plusieurs clusters. Les affectations les moins probables sont écartées, supprimant efficacement le bruit. Le MAC produit une suppression de bruit quasi parfaite jusqu'à des niveaux de bruit de 40% évalués sur des données synthétiques utilisant un modèle de bruit par inversion de bit. Cette approche obtient aussi de meilleurs résultats en termes de stabilité de solution que d'autres méthodes générales de suppression de bruit sur matrices binaires : Infinite Noisy-OR (INO) [30] une approche non supervisée probabiliste, Discrete Basis Problem solver (DBPs) [31] un algorithme glouton de décomposition matricielle, et Binary Independent Component Analysis (BICA) [32] une méthode variationnelle de factorisation et suppression de bruit spécifique pour les matrices binaires.

D'autres approches préfèrent un processus en 2 étapes : nettoyer les données d'abord et miner ensuite. L'approche de Molloy et al. [24] utilise des algorithmes de décomposition de matrice binaire pour l'étape de nettoyage. Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), Binary Non-Negative Matrix Factorization (BNMF) et logistic Principal Component Analysis (PCA) en tant qu'algorithmes servant au nettoyage des données, adjoints d'un autre algorithme de RM exprimant la matrice UPA de façon exacte. Cette approche est comparée au δ-RMP, DDM et MAC. La matrice UPA est d'abord décomposée, puis reconstruite en format binaire en utilisant la transformation inverse de la décomposition utilisée. Une fonction échelon assure des valeurs binaires lors de la reconstruction. Puisque les décompositions ne sont pas exactes, certaines données sont détruites, supprimant ainsi le bruit. Les rôles sont ensuite minés en utilisant deux algorithmes exacts : HPr [1] et HM [20]. L'évaluation de Molloy et al. [24] démontre que la méthode en 2 étapes produit de meilleurs résultats de suppression de bruit sur des jeux de données synthétiques que les autres approches mentionnées (δ-RMP, DDM, MAC).

2.3 Évaluation

L'évaluation des performances des algorithmes de role mining nécessite l'utilisation de jeux de données standards ou couramment utilisés qui sont référencés dans la plupart des articles, ou du moins des jeux synthétiques bien définis. Les jeux de données du monde réel sont généralement utilisés pour évaluer les approches de role mining classiques qui opèrent sur des données propres, car elles sont évaluées sur leurs performances brutes (temps d'exécution, complexité de la hiérarchie de rôles produite, etc.). Cependant, des jeux de données synthétiquement bruités sont nécessaires pour évaluer les performances des approches conçues pour traiter des données comportant des anomalies. De plus, les métriques utilisées pour l'évaluation doivent être adaptées pour prendre en compte cette contrainte additionnelle.

2.3.1 Jeux de données réels

Les jeux de données réels provenant de configurations de systèmes de contrôle d'accès sont rares, ils sont donc fréquemment fournis par un partenaire industriel lorsque cela est possible, sans pour autant être divulgués. Cependant, certains jeux de données ont été rendus publics.

Ene et al. [1] aux côtés de membres d'Hewlett-Packard Labs ont publié un article sur des méthodes exactes fondées sur des graphes utilisant des heuristiques pour aborder le problème de role mining de manière plus efficace. Une des contributions notables de cet article est la publication de jeux de données réels rendus disponibles par Hewlett Packard Labs. En raison de la rareté des jeux de données réels de contrôle d'accès, ceux-ci sont rapidement devenus la référence pour évaluer les algorithmes de role mining, et plus largement les approches de GIA [33–43]. Les jeux de données sont les suivants :

- **americas_large** et **americas_small** qui proviennent de pare-feux Cisco authentifiant et autorisant des utilisateurs externes sur le réseau interne de HP.
- **apj** et **emea** proviennent aussi de pare-feux, mais sont de taille moindre.
- **healthcare** provient de l'United States Veteran's Administration. C'est une liste compréhensive des permissions à assigner aux professionnels de santé agréés.
- **domino** est le jeu de données le plus proche d'une Access Control List (ACL), c'est un jeu de données de profils d'accès utilisateur d'un serveur Lotus Domino.
- **customer** provient à l'origine un graphe de contrôle d'accès du département informatique d'un client Hewlett-Packard.
- **firewall_1** et **firewall_2** sont des jeux de données qui résultent d'un algorithme d'analyse sur des pare-feux Check Point. L'analyse en question teste l'accessibilité sur des services (c'est-à-dire SSH, HTTP ...) pour différents utilisateurs.

Leurs dimensions sont renseignées sur la table 2.1.

Jeu de données	Utilisateurs	Permissions	Assignations
americas_large	3,485	10,127	185,294
americas_small	3,477	1,587	105,205
apj	2,044	1,164	6,841
emea	35	3,046	7,220
healthcare	46	46	1,486
domino	79	231	730
customer	10,021	277	45,427
firewall 1	365	709	31,951
firewall 2	325	590	36,428

Table 2.1 Résumé des dimensions des jeux de données HP Labs [1]

2.3.2 Jeux de données synthétiques

Les jeux de données synthétiques fournissent un cadre flexible et précis pour évaluer les algorithmes de RM. La génération est faite avec un objectif spécifique de RM, qui est ensuite dérivé pour construire la matrice UPA. Les jeux de données synthétiques permettent donc une comparaison directe entre la sortie d'un algorithme de RM et l'objectif sous-jacent utilisé pour construire le jeu de données. Cette approche assure une évaluation plus ciblée et précise de la performance d'un algorithme.

Une des premières méthodes utilisées pour générer de tels jeux de données fut inventée par Vaidya et al. [19]. Elle est aussi connue sous le nom de Random Data Generator [33]. Ce générateur fonctionne en 3 étapes :

1. Un ensemble de rôles est créé avec un nombre aléatoire de permissions attribuées entre 1 et un maximum. Le nombre maximum de permissions par rôle est un paramètre de l'algorithme

2. Un ensemble d'utilisateurs est créé avec un nombre aléatoire de rôles entre 0 et un maximum. Encore une fois, le nombre maximum de rôles qu'un utilisateur peut recevoir est un paramètre de l'algorithme.

3. La matrice UPA est générée en traduisant Pour chaque utilisateur, les permissions héritées des rôles qui leur ont été attribués.

Essentiellement, cette méthode construit les matrices UA et PA pour en dériver ensuite la matrice UPA. La méthode fournit un objectif de role mining clair à retrouver bien qu'il ne

soit pas minimal. Dans beaucoup de cas, il est possible de trouver un ensemble de rôles plus petit que celui utilisé lors de la génération pour exprimer la matrice UPA, notamment dans le cas où des rôles sont contenus dans d'autres et attribués aux mêmes utilisateurs. Cependant, cette méthode est encore utilisée pour générer des jeux de données même très récemment [39] [44], car elle permet une comparaison relative d'algorithme à algorithme.

Molloy et al. [33] proposent deux autres schémas de génération de données : Tree et Enterprise Role Based Access Control (ERBAC). Avec le Tree Data Generator, le jeu de données est généré pour imiter une organisation divisée en départements, sont divisés eux même en bureaux, etc. Étant donné cette hypothèse, l'algorithme construit d'abord un arbre avec une hauteur définie et des nœuds avec un nombre borné d'enfants. Ensuite, il assigne des permissions à chaque nœud. Les enfants héritent des permissions de leurs parents, imitant des permissions à l'échelle de l'organisation, du département, du bureau, etc. Cet arbre représente les permissions possibles dans la structure sous-jacente. Pour produire le jeu de données final, les utilisateurs sont assignés aux feuilles de l'arbre et leurs permissions sont choisies dans l'ensemble réduit de permissions données par l'arbre et assignées en utilisant le Random Data Generator. Une illustration de la structure formée par Tree est renseignée sur la figure 2.2.

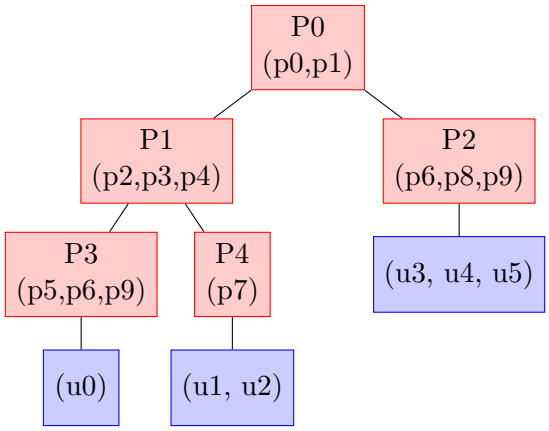


Figure 2.2 Exemple de structure générée par Tree

Ici l'utilisateur u1 peut hériter d'un sous ensemble des permissions contenues dans les nœuds P0, P1 et P4, par exemple (p1, p2, p4, p7)

Avec le Générateur de Données ERBAC, le jeu de données est généré en s'inspirant du modèle Enterprise RBAC conceptualisé par Kern et al. [27] pour aider à déployer les systèmes RBAC. Ce modèle fut développé avec des insights pratiques obtenus grâce au déploiement de systèmes RBAC dans des scénarios du monde réel. Le modèle requiert une hiérarchie de rôles à deux couches : rôles fonctionnels et rôles métier. Chaque permission assignée à un nombre aléatoire de rôles fonctionnels, les rôles métier fédèrent plusieurs rôles fonctionnels, et chaque utilisateur

reçoit plusieurs rôles métier. Les règles d'héritage confèrent aux utilisateurs les permissions rattachées aux rôles fonctionnels de leurs rôles métier. L'algorithme fonctionne donc en quatre étapes : d'abord les rôles fonctionnels reçoivent un nombre aléatoire de permissions, deuxièmement chaque rôle métier reçoit un nombre aléatoire de rôles fonctionnels, puis chaque utilisateur reçoit un nombre aléatoire de rôles métier, finalement la matrice UPA est calculée avec les règles d'héritage. Une illustration de la structure générée par ERBAC est renseignée sur la figure 2.3.

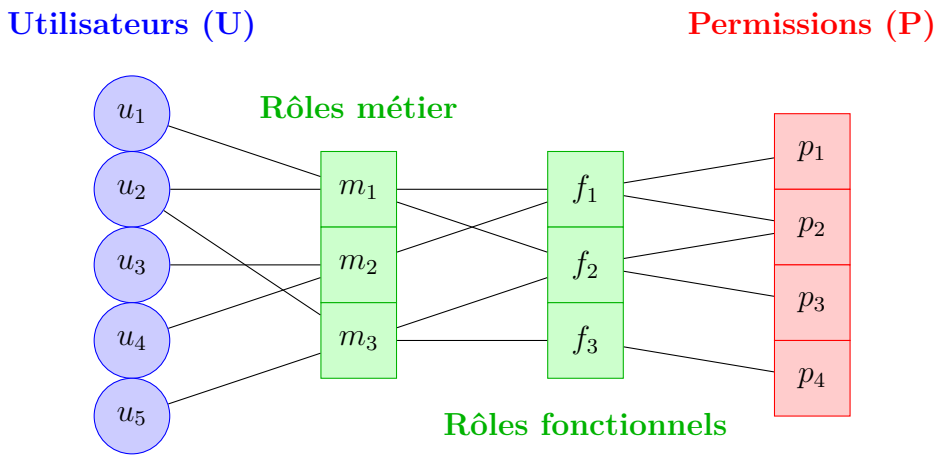


Figure 2.3 Exemple de structure générée par ERBAC Data Generator

Ici u5 hérite des permissions p2, p3 et p4

Comme nous pouvons le voir, ces deux générateurs utilisent une certaine abstraction pour construire un jeu de données qui ressemblerait davantage à un jeu de données du monde réel. Cependant, les trois premières méthodes de génération mentionnées ne prennent pas en compte le problème de l'accumulation des privilèges. Les jeux de données produits sont ensuite bruités aléatoirement avec des assignations supplémentaires, ne respectant donc pas de scénario réel. [19, 24, 33]. Les jeux de données sont alors bruités de manière uniformément aléatoire ou avec un processus avec une loi de Bernoulli avec une probabilité faible. Les jeux de données produits sont alors en désaccord avec la réalité rencontrée dans le milieu des entreprises, bien que les permissions légitimes soient "bien formées". Aussi, l'état partiel utilisé pour construire la matrice UPA, contenant toute l'information hiérarchique, n'est pas exploitée pour construire un objectif de role mining intéressant.

Une autre méthode par Abolfathi et al. [45] utilise des modèles administratifs par Stoller et al. [46] pour construire une matrice UPA crédible. Ce générateur produit un état RBAC basé sur un modèle de contrôle d'accès en milieu universitaire. Il inclut les rôles typiques présents dans un système universitaire comme étudiant, assistant d'enseignement, doyen, directeur de programme d'honneur, etc. Il inclut aussi une hiérarchie de rôles imitant la vraie hiérarchie de l'université. Le générateur prend alors un nombre donné d'utilisateurs et de permissions et génère un état RBAC en attribuant les utilisateurs et les permissions sur le template avant d'utiliser les règles d'héritage et d'attribution de rôles pour produire la matrice UPA. Cette méthode est celle qui produit des jeux de données les plus réalistes dans un cadre universitaire, mais perd en généralité si un autre template n'est pas proposé. De plus, l'approche par Abolfathi et al. produit des jeux de données sans bruit ou accumulation de privilèges par sa conception.

Parkinson et al. [47] proposent une approche pour introduire synthétiquement de l'accumulation de privilèges dans des jeux de données de contrôle d'accès de systèmes de fichiers. L'algorithme génère le jeu de données de système de fichier itérativement puis ajoute des permissions relatives à un utilisateur et un répertoire dans le système de fichiers. Ces permissions supplémentaires nommées "permission creep" sont directement données aux utilisateurs (et non à leur groupe) par sélection pseudo-aléatoire. L'allocation est ensuite appliquée au système de fichiers, et est également écrite dans un fichier texte pour être utilisée comme connaissance de référence pour évaluer la performance de détection de l'algorithme développé. Puisque ces permissions sont introduites de façon isolée sur des utilisateurs spécifiques et non aléatoirement sur l'ensemble des utilisateurs, cette méthode d'introduction d'anomalies est la plus réaliste rencontrée jusqu'à présent. Cependant, les permissions additionnelles données lors de l'attribution de l'accumulation de privilèges sont choisies dans l'ensemble total des permissions, ce qui peut être une hypothèse forte.

L'accumulation de privilèges est aussi désignée dans la littérature sous les termes "permission creep" [47] ou "privilege creep" [48]. Par la suite dans ce mémoire, on désigne toute anomalie qui relève de l'accumulation de privilèges par le terme "privilege creep" avec l'acronyme PC, pour simplifier la terminologie et faciliter la lecture.

2.4 Métriques

Les premières métriques utilisées pour évaluer les algorithmes de role mining exacts utilisent généralement la sortie d'un algorithme directement. Le nombre de rôles produits et le temps d'exécution sont quasi systématiquement rapportés, car ils permettent une interprétation directe [19, 33, 38, 39, 42, 43, 45, 46, 49–55].

Pour généraliser les métriques de minimisation et d'évaluation de plusieurs approches, Molloy et al. [49] ont introduit une métrique paramétrée appelée Weighted Structural Complexity (WSC) ou Complexité Structurelle Pondérée (CSP) :

Soit $W = \langle w_r, w_u, w_p, w_h, w_d \rangle$ un vecteur de poids où chaque composante $w_r, w_u, w_p, w_h, w_d \in \mathbb{Q}^+ \cup \{\infty\}$ représente un paramètre de pondération pour la CSP. Ils correspondent respectivement au poids sur le nombre de rôles, le nombre d'assignations utilisateur-rôle, le nombre d'assignations rôle-permission, la complexité de la hiérarchie de rôles et le nombre d'assignations directes. Alors la CSP d'un état RBAC γ est notée $\text{wsc}(\gamma, W)$, et est calculée comme suit :

$$\text{wsc}(\gamma, W) = w_r \cdot |R| + w_u \cdot |UA| + w_p \cdot |PA| + w_h \cdot |t\_reduce(RH)| + w_d \cdot |DUPA| \quad (2.2)$$

Où :

- $|R|$ est le nombre de rôles de l'état RBAC
- $|UA|$ et $|PA|$ sont les normes de Manhattan de leur matrice correspondante
- $|t\_reduce(RH)|$ est le nombre minimal de relations qui traduit la hiérarchie des rôles (réduction transitive). Un bon exemple est donné dans l'article avec une hiérarchie simple : $t\_reduce(\{(r_1, r_2), (r_2, r_3), (r_1, r_3)\}) = \{(r_1, r_2), (r_2, r_3)\}$, car (r_1, r_3) peut être inféré. On s'en sert de mesure de la complexité hiérarchique de l'état RBAC.
- |DUPA| est le nombre d'assignations directes ou exceptions qui peuvent survenir lorsque la matrice UPA n'est pas entièrement reconstruite avec les matrices UA et PA. On rajoute donc manuellement ces assignations hors RBAC.

Comme Molloy et al. [33] l'indiquent dans un autre article, plusieurs objectifs RMP peuvent être dérivés si on minimise la CSP :

- BasicRMP [22] est atteint avec $W = \langle 1, 0, 0, 0, \infty \rangle$
- La variante de Vaidya [18] d'edge-RMP est atteinte avec $W = \langle 0, 1, 1, 0, \infty \rangle$

- La variante de Zhang [16] d'edge-RMP $W = \langle 1, 1, 1, 0, \infty \rangle$
- Une forme de δ-cohérence [22] peut être atteinte avec $W = \langle x_1, x_2, x_3, x_4, 1 \rangle$ pour tout (x_1, x_2, x_3, x_4), par exemple $W = \langle 1, 0, 0, 0, 1 \rangle$ est similaire au δ-RMP, minimisant les rôles et les affectations directes d'utilisateurs.
- $W = \langle 1, 1, 1, 1, 1 \rangle$ est le vecteur proposé par Molloy et al. [33] pour comparer différents algorithmes sur la minimisation de la complexité RBAC en autorisant les assignations directes. L'intérêt de ce vecteur de poids est de pénaliser les algorithmes qui vont surajuster (overfit) le jeu de données en produisant une hiérarchie de rôles trop précise, réduisant alors l'avantage de gestion offert par RBAC.

Pour les approches de role mining inexactes qui visent une gestion simplifiée de l'état RBAC, on compare généralement l'entrée UPA et la sortie UA et PA de l'algorithme de role mining utilisé. On peut alors définir la distance de Jaccard, l'erreur de reconstruction, etc.

La distance de Jaccard J_δ compte le nombre de dissimilarités entre deux matrices booléennes. Dans notre cas, elle est calculée comme suit :

$$J_\delta(UPA, UA \otimes PA) = 1 - \frac{\sum_{u,p} UPA_{u,p} \wedge (UA \otimes PA)_{u,p}}{\sum_{u,p} UPA_{u,p} \vee (UA \otimes PA)_{u,p}} \tag{2.3}$$

Où $\wedge$ représente le ET logique et $\vee$ le OU logique.

L'erreur de reconstruction $E_{reconstruction}$ peut alors être exprimée en utilisant la distance de Jaccard entre la matrice d'origine UPA et la matrice reconstruite $UA \otimes PA$:

$$E_{reconstruction}(UPA, UA \otimes PA) = \frac{J_\delta(UPA, UA \otimes PA)}{|UPA|} \tag{2.4}$$

Ces métriques servent principalement pour évaluer la performance d'algorithmes qui prennent en compte le bruit d'applicabilité RBAC afin de simplifier la gestion de l'état RBAC, mais elles ne sont pas utilisées dans le cadre de role mining sensible au privilege creep pour en évaluer la détection [39, 50, 51, 55, 56].

Vaidya et al. [23] introduit la notion de robustesse au bruit (noise robustness) pour les algorithmes de role mining : la robustesse au bruit reflète à quel point un algorithme de RM est affecté ou non par le bruit lorsqu'il opère sur un jeu de données. Vaidya et al. identifient qu'une définition appropriée pour le degré de robustesse au bruit devrait prendre en compte à la fois le bruit supprimé efficacement, et les erreurs commises par l'algorithme. On a alors besoin d'une métrique qui varie positivement avec le pourcentage de bits bruités correctement

reconstitués et négativement avec le pourcentage d'erreurs commises par l'algorithme. Vaidya et al. utilisent donc la F-mesure.

Il est important de comprendre pourquoi cette mesure est appropriée. D'abord, il faut définir deux classes sur les bits de la matrice UPA :

- Les bits dits originaux, ou vrais bits qui n'ont pas été inversés par le bruit.
- Les bits dits inversés, ou bits bruités qui doivent être corrigés par l'algorithme

On définit ensuite :

- Les vrais positifs (TP) : les bits bruités que l'algorithme a correctement identifiés et corrigés pour les ramener à leur état original.
- Les faux positifs (FP) : les vrais bits que l'algorithme a incorrectement modifiés (a introduit des erreurs là où il n'y avait pas de bruit).
- Les vrais négatifs (TN) : les vrais bits que l'algorithme a correctement laissés inchangés.
- Les faux négatifs (FN) : les bits bruités que l'algorithme n'a pas réussi à corriger (a laissé le bruit non corrigé).

Pour construire la F-mesure, il faut définir le rappel et la précision en tant que métriques.

Le rappel mesure la complétude : comment les bits bruités ont été corrigés par l'algorithme. Une valeur de rappel élevée signifie que la plupart des bits bruités ont été corrigés avec succès :

$$Rappel = \frac{TP}{TP + FN} \tag{2.5}$$

La précision mesure l'exactitude/fidélité : De tous les bits qui ont été inversés par l'algorithme, combien en avaient réellement besoin d'être inversés ?. Une valeur de précision élevée reflète que l'algorithme n'a pas introduit de corrections inutiles :

$$Précision = \frac{TP}{TP + FP} \tag{2.6}$$

Enfin, la F-mesure est la moyenne harmonique de la précision et du rappel :

$$F = \frac{2 \times Précision \times Rappel}{Précision + Rappel} \tag{2.7}$$

Avec cette définition, il devient apparent que le choix de Vaidya et al. pour évaluer la performance d'un algorithme de role mining, demande un processus de génération de jeu de données synthétiquement bruité, auquel on a accès à l'étape intermédiaire sans bruit.

Enfin Parkinson et al. [47] proposent une métrique simple pour évaluer la performance en détection des instances de privilege creep, en définissant l'Exactitude (accuracy) comme suit :

$$Exactitude = \frac{tpr + tnr}{tpr + tnr + fpr + fnr} = \frac{tpr + tnr}{2} \tag{2.8}$$

Avec :

- Taux de vrais positifs (tpr) ou Sensibilité : la fraction de permissions de privilege creep correctement identifiées comme faisant partie d'une instance de privilege creep
- Taux de faux positifs (fpr=1-tnr) : la fraction des permissions régulières incorrectement identifiées comme faisant partie d'une instance de de privilege creep
- Taux de vrais négatifs (tnr) ou Spécificité : la fraction des permissions régulières correctement identifiées comme régulières
- Taux de faux négatifs False Negative Rate (fnr=1-tpr) : la fraction des permissions de privilege creep incorrectement classifiées comme régulières.

On voit donc que la formule de Parkinson est la moyenne arithmétique de la sensibilité et la spécificité de la détection des permissions de privilege creep.

Enfin une dernière métrique importante à mentionner est celle du score d'interprétabilité de Kang et al. [55]. Cette métrique quantifie à quel point un rôle est interprétable en fonction des attributs (hors permissions) des utilisateurs qui le portent. Plus les utilisateurs portant un même rôle ont des attributs similaires, meilleur est le score pour ce rôle. Le score d'interprétabilité d'un état RBAC est alors défini comme la somme arithmétique des scores d'interprétabilité de tous les rôles. Il nécessite donc la présence d'attributs pertinents et exploitables. Cette métrique est formalisée comme suit :

On définit l'expression d'un attribut toute règle qui décrit un profil type d'utilisateur, par exemple "département=Finance ET ancienneté>5ans".

Le décalage MM (mismatch) entre une expression d'attribut e et un ensemble U d'utilisateurs est défini par l'équation suivante :

$$\mathrm{MM}(e, U) = |(U_e \setminus U) \cup (U \setminus U_e)| \tag{2.9}$$

Où U^e est l'ensemble des utilisateurs qui satisfont e.

L'interprétabilité d'un rôle est ensuite exprimée comme le degré de décalage entre l'ensemble des utilisateurs portant ce rôle assignU(r) et l'ensemble des expressions d'attribut e. Plus le

degré de décalage est faible, plus l'interprétabilité du rôle est forte. L'interprétabilité d'un rôle r nommée AMM(r), définie dans l'équation (2.10), dénote le décalage minimum atteint par le rôle r.

$$\mathrm{AMM}(r) = \min_{e \in E}(\mathrm{MM}(e, \mathrm{assignU}(r))) \tag{2.10}$$

Où E est l'ensemble de toutes les expressions d'attributs, $\mathrm{assignU}(r) = \{u \mid \langle u, r\rangle \in UA\}$ représente l'ensemble des utilisateurs auxquels ce rôle est assigné. L'interprétabilité d'une politique RBAC INT est alors :

$$\mathrm{INT} = \sum_{r \in R} \mathrm{AMM}(r) \tag{2.11}$$

Les avantages et inconvénients des différentes métriques d'évaluation utilisées dans la littérature sont compilées dans le tableau 2.2. Dans le cadre de ma recherche, les inspirations prises de ces métriques sont renseignées dans la section 5 dans le tableau 5.4.

2.5 Approches récentes

2.5.1 Méthodes de role mining

Durdag et Coskuncay [56] proposent une méthode pour reconfigurer les systèmes RBAC spécifiques aux clients en regroupant les rôles similaires selon la similarité des permissions. Leur approche utilise le Agglomerative Hierarchical Clustering (AHC) en utilisant la distance de Jaccard sur des données RBAC réelles collectées auprès de dix clients d'une entreprise de logiciels. Plutôt que de viser à remplacer directement les rôles existants, cette méthode identifie des groupes de rôles qui servent de structures de référence pour soutenir la refonte ou le nettoyage du système. Les résultats du clustering sont évalués à l'aide de la précision, du rappel et de F-mesure. Les résultats montrent que les dendrogrammes produits produisent des regroupements de rôles cohérents qui servent efficacement l'objectif de refonte de système, améliorant l'interprétabilité et la gestion des configurations RBAC complexes. La méthode proposée ne prend pas en compte la présence de privilege creep, bien que des données réelles issues d'entreprise soient utilisées.

Un article de Zhu et al. [43] apporte une modification aux solutions du edge-RMP. En effet, plusieurs algorithmes pour traiter ce problème n'affinent pas assez les affectations dans les matrices UA et PA. De plus, lors du traitement de grands ensembles de données, les algorithmes préexistants produisent beaucoup de rôles redondants. Il est donc essentiellement question d'améliorer la performance d'algorithmes basés sur les graphes en proposant un algorithme avec plus d'optimisations. La solution proposée ne prend donc pas explicitement en compte le

privilege creep. Cependant, les résultats expérimentaux démontrent que l'algorithme amélioré diminue efficacement les coûts de gestion tout en fournissant des temps d'exécution plus courts.

Kang et al. [55] améliorent les algorithmes précédents de RM bruité en introduisant la notion d'interprétabilité des rôles. Chaque rôle extrait se voit attribuer un score d'interprétabilité basé sur les attributs des utilisateurs à qui on affecte ce rôle. Plus les attributs des utilisateurs portant un même rôle sont similaires, plus le score d'interprétabilité est bas. Les rôles sont ensuite sélectionnés afin de minimiser ce score. L'erreur de reconstruction est ensuite calculée entre la matrice UPA originelle et la matrice minée pour évaluer le degré de permissions non exprimées. Cette approche est ensuite évaluée avec des jeux de données synthétiques générées avec le Random Data Generator et sur des jeux de données réels notamment *firewall1* et *healthcare* [1]. Cette méthode prend en compte la notion de bruit, comme source de complexification des jeux de données, mais pas celle de privilege creep en tant qu'anomalie à identifier et supprimer.

Nobi et al. [57] introduisent une nouvelle technique : Deep Learning Based Access Control (DLBAC). Cette méthode est fondée sur des réseaux de neurones qui apprennent directement à partir des métadonnées brutes des utilisateurs et des ressources. Le prototype DLBAC_α démontre une précision et une généralisation supérieures par rapport aux approches classiques de role mining et d'apprentissage automatique, tout apportant une contribution additionnelle : l'explicabilité, grâce à des techniques d'interprétation comme les "integrated gradients". Cependant, DLBAC ne traite pas directement le problème d'accumulation de privilèges, car il apprend à partir de données d'autorisation existantes qui peuvent déjà contenir des droits d'accès anomaux. Cette limitation est reconnue par les auteurs, qui indiquent que des erreurs dans les jeux de données utilisés pourraient introduire un biais dans le modèle entraîné.

2.5.2 Méthodes de détection de privilege creep

Parkinson et al. [47] présentent un outil non supervisé pour détecter les instances de privilege creep dans les ACL de système de fichiers Microsoft NT. L'approche utilise une méthode statistique : dans ce cadre, une irrégularité statistique dans l'ensemble des assignations de permissions est considérée comme une instance de privilege creep. Les outils utilisés, l'analyse χ^2 et les ruptures naturelles de Jenks, permettent d'identifier automatiquement ces anomalies en détectant respectivement les écarts significatifs par rapport à une distribution attendue et les discontinuités naturelles dans les données d'assignation de permission. Cette méthode est ensuite implémentée en C# sous le nom de "Creeper". Des tests empiriques sont réalisés sur des jeux synthétiques contenant différents niveaux de privilege creep synthétiquement

ajoutés. Une autre analyse empirique est ensuite réalisée sur 5 systèmes réels pour établir la précision de Creeper par le biais d'une étude comparative entre un expert manuel (humain), ntfs-r [52], et Creeper. La précision est ici définie comme la fraction de tous les échantillons correctement identifiés. L'évaluation démontre que Creeper atteint une précision moyenne de 96% sur les jeux de données synthétiques, et une précision moyenne de 98% sur les systèmes réels. De plus, l'analyse sur systèmes réels a démontré une amélioration significative de la précision par rapport à deux autres techniques : expert manuel et ntfs-r. L'outil Creeper fait partie des rares outils récents développés en role mining qui prennent en compte l'enjeu du privilege creep dans les systèmes de contrôle d'accès. Bien que l'approche ne soit pas directement un algorithme ou une méthode de role mining, la génération synthétique des données faites dans cet article s'appuie sur une structure RBAC afin de créer le système de fichier sur lequel l'approche de détection est évaluée. Parkinson et al. [58] proposent également une approche basée sur la "fuzzy logic" pour identifier l'accumulation critique de privilèges dans les politiques de contrôle d'accès en modélisant la confiance des utilisateurs, la sensibilité des ressources et la puissance des permissions comme des ensembles flous plutôt que comme des classifications binaires. Cette nouvelle approche s'appuie sur les journaux d'événements de sécurité, produisant de meilleurs résultats que l'outil Creeper.

Alexander et Chikwarti [48] proposent un cadre d'intelligence artificielle basé sur des graphes modélisant l'Identity and Access Management (IAM) d'entreprise comme un graphe de connaissances. Ils appliquent des réseaux de neurones graphiques (GNN), des algorithmes d'extraction de communautés (méthode de Louvain) et du clustering de graphes pour découvrir des structures de rôles latentes à partir des modèles d'accès. Pour détecter les permissions anormales, des autoencodeurs sont utilisés. Leur approche prétend réduire la redondance des rôles de 38% et atteindre une précision de 93,5% dans la détection d'anomalies. Cependant, l'évaluation a été faite exclusivement sur des ensembles de données synthétiques et la comparaison d'algorithme a été faite avec une référence inappropriée utilisant K-means, inadaptée pour la détection d'anomalies, rendant ainsi l'amélioration rapportée de la F-mesure de 74,8 % à 91,3 % potentiellement trompeuse.

2.5.3 Méthodes de détection d'anomalies générales

Dans d'autres domaines scientifiques, utiliser la réduction de dimensionnalité aux côtés du clustering permet la détection d'anomalies tout en produisant des groupes avec des attributs semblables. Cette approche pourrait être utilisée en RM pour supprimer le bruit, détecter les anomalies d'utilisateurs et guider l'effort de construction des rôles.

Dans un article sur la détection de fraude [59], Massi et al. proposent une approche en deux

étapes pour détecter la fraude en santé sur une base de données administrative hospitalière. La première étape est la sélection de caractéristiques utilisant Principal Feature Analysis (PFA) de Lu et al. [60]. La seconde étape est l'application de l'algorithme K-Means sur les caractéristiques sélectionnées pour regrouper ensemble les hôpitaux similaires. Cette méthode a efficacement identifié les outliers et a même indiqué les caractéristiques significatives qui rendaient certains hôpitaux "anormaux", et les a donc signalés comme frauduleux.

Souza et al. [61] utilisent une approche semblable en deux étapes pour détecter les outliers des espaces urbains intelligents. High-Order Singular Value Decomposition (HOSVD) [62] est utilisée dans la première étape pour déterminer les composantes significatives, puis le clustering est effectué sur les composantes utilisant K-means. Bien que la méthode nécessite un grand nombre d'échantillons pour détecter les outliers, elle identifie les motifs spatio-temporels sur les espaces urbains.

2.5.4 Lacunes

Comme le montre la revue de littérature, les approches de role mining, même récentes, se concentrent principalement sur des jeux de données ou le privilege creep n'est pas présent. La première amélioration possible est donc de prendre en compte le privilege creep comme étant un phénomène présent dans les systèmes de contrôle d'accès qu'il faut retirer pour éviter de les répéter en RBAC.

On pourrait imaginer que le bruit en tant qu'anomalie peut être considéré comme une instance de privilege creep. Or, beaucoup d'articles font le choix de définir le bruit comme uniformément aléatoire. Cet ajout gêne en partie l'effort de role mining, mais ne correspond pas à un scénario crédible où du privilege creep pourrait avoir lieu, et ne correspond pas à la réalité du privilege creep dans une organisation possédant déjà une infrastructure RBAC. En effet, on devrait voir des ensembles de permissions reliés à d'anciens rôles qu'on a oublié de révoquer, et non des permissions isolées. Une seconde amélioration serait donc de modifier la génération de jeux de données synthétiques de sorte à injecter des instances de privilege creep d'une nouvelle forme en plus du bruit.

Enfin, la littérature du role mining manque d'un cadre pour l'évaluation des approches qui prennent en compte le privilege creep. Ce qui s'en rapproche le plus est la métrique de noise robustness de Vaidya et al. [23] et la métrique d'exactitude de Parkinson et al. [47, 52, 58]. Cependant, la métrique de noise robustness n'est pas appropriée pour évaluer correctement la détection ou la correction de privilege creep étant donné que le bruit est, lui aussi, présent dans les jeux de données réels. Aussi, la métrique d'exactitude est biaisée par le fait qu'il y a un déséquilibre entre les permissions légitimes et les permissions issues du privilege creep :

dans un système typique, 90 à 95% des permissions sont légitimes tandis que 5 à 10% au plus peuvent être attribués à du privilege creep.

En prenant un exemple concret pour comprendre le problème, imaginons un système avec 10,000 assignations : 9,500 légitimes et 500 anormales issues de privilege creep. L'algorithme évalué détecte 450 vraies instances de privilege creep et 1,000 fausses alertes. On considère normalement ce genre de sortie comme produisant trop de faux positifs.

La sensibilité/rappel = 450/(450+50) = 0,90, La spécificité = 8,500/(8,500+1,000) = 0,89 semble bonne, mais la précision = 450/(450+1,000) = 0,31 révèle une performance moindre au vu du nombre écrasant de faux positifs. L'exactitude donnerait donc une valeur de 0,9 tandis que la F-mesure donnerait 0,46.

Tout ces éléments porte à croire qu'il y a besoin d'un cadre plus pertinent afin d'évaluer la détection de privilege creep dans le domaine du role mining. D'où une partie de la contribution finale de ma recherche qui est d'adapter le cadre d'évaluation, dont les métriques sont résumées dans le tableau 2.2, pour tenir compte du privilege creep.

Métrique	**Avantages et Inconvénients**
Nombre de rôles ($\|R\|$) et temps d'exécution	**Avantages :** Métriques directes et simples, faciles à mesurer, permettent une interprétation immédiate de la performance. Bonne métrique de comparaison **Inconvénients :** Ne considèrent pas la qualité des rôles produits, le temps d'exécution dépend de l'implémentation et du matériel, le nombre de rôles peut être plus grand pour une meilleure gestion des accès.
Complexité Structurelle Pondérée (CSP) d'un état RBAC	**Avantages :** Métrique paramétrable très flexible, permet de cibler différents objectifs RMP selon les poids choisis, couvre tous les aspects d'un état RBAC. **Inconvénients :** Nécessite de définir les poids appropriés selon le contexte, peut être complexe à interpréter, ne prend pas en compte la qualité sémantique des rôles.
Distance de Jaccard (J_δ) entre entrée et sortie d'algorithme	**Avantages :** Simple à calculer, mesure directement la similarité entre matrices, intuitive à interpréter. **Inconvénients :** Ne distingue pas les types d'erreurs (faux positifs vs faux négatifs), sensible à la taille des matrices, ne considère que l'aspect binaire des permissions.
Erreur de reconstruction ($E_{reconstruction}$) entre entrée et sortie d'algorithme	**Avantages :** Normalise la distance de Jaccard par la taille de la matrice, permet la comparaison entre différentes tailles de systèmes. **Inconvénients :** Hérite des limitations de la distance de Jaccard, ne fournit qu'une vue globale sans détails sur le type d'erreur. Inutile dans le cadre de role mining avec privilege creep car cela gonfle l'erreur de reconstruction si les permissions problématiques sont retirées
F-mesure pour le calcul de robustesse au bruit	**Avantages :** Équilibre précision et rappel, appropriée pour évaluer la robustesse au bruit, métrique standard en classification. **Inconvénients :** Nécessite des données de référence (ground truth), plus complexe à calculer.
Exactitude pour la détection de privilege creep	**Avantages :** Simple à comprendre, moyenne équilibrée de sensibilité et spécificité, métrique intuitive pour la détection de privilege creep. **Inconvénients :** Nécessite des données de référence (ground truth), peut être biaisée si les classes sont déséquilibrées.
Interprétabilité (INT) de politique RBAC	**Avantages :** Évalue la cohérence sémantique des rôles, prend en compte les attributs utilisateur, favorise des rôles compréhensibles pour les administrateurs. **Inconvénients :** Nécessite des attributs utilisateur pertinents et exploitables, complexe à calculer, dépendant de la qualité des expressions d'attributs définies.

Table 2.2 Récapitulatif des métriques d'évaluation d'algorithmes de role mining

CHAPITRE 3 ROLE MINING SENSIBLE AU PRIVILEGE CREEP

Cette partie détaille l'approche de role mining sensible du privilege creep. Elle est organisée autour des étapes principales suivantes : les prérequis, qui présentent les concepts de réduction de dimension et de clustering ainsi que le prétraitement des données ; la réduction de dimension, permettant de capturer les motifs principaux dans la matrice UPA ; le clustering et l'identification des outliers, pour détecter les utilisateurs aux permissions atypiques ; le nettoyage, qui prépare la matrice pour l'extraction fiable des rôles ; le role mining, étape où les rôles sont extraits et assignés aux utilisateurs ; et enfin la réunification, qui réintègre les outliers selon les approches omnisciente et heuristique.

3.1 Prérequis

3.1.1 Introduction des concepts de réduction de dimension et de clustering

Tous les concepts d'algèbre linéaire introduits par la suite opèrent dans notre cas sur des matrices binaires sans perte de généralité.

SVD et Truncated Singular Value Decomposition (TSVD)

En mathématiques, la décomposition en valeurs singulières (SVD) est un procédé d'algèbre linéaire de factorisation de matrices rectangulaires. Il peut être utilisé pour des calculs de matrice pseudo inverse, traitement automatique des langues, etc. Dans le cadre de cette recherche cependant, la SVD permet de créer des approximations de rang faible des matrices UPA, réduisant ainsi la dimensionnalité des données tout en préservant les caractéristiques les plus importantes. Cette propriété d'approximation est particulièrement utile pour la réduction de bruit, et l'extraction des motifs principaux dans des ensembles de données de grande dimension.

Soit M une matrice binaire $m \times n$ dont les coefficients appartiennent à $\{0, 1\} \subset \mathbb{R}$. Bien que M soit binaire, sa décomposition en valeurs singulières s'écrit sous la forme classique :

$$M = U\Sigma V^T \tag{3.1}$$

avec :

- U une matrice orthogonale $m \times m$ sur $\mathbb{R}$.
- Σ une matrice $m \times n$ dont les coefficients diagonaux (σ_i) sont des réels positifs ou nuls et tous les autres sont nuls.
- V^T est la matrice transposée de V, matrice orthogonale $n \times n$ sur $\mathbb{R}$.

Interprétation des composantes :

Vecteurs d'analyse (V) : Chaque colonne de V représente un "profil type" ou une combinaison linéaire des variables binaires originales. Dans le contexte d'une matrice binaire, ces vecteurs révèlent les associations sous-jacentes entre les variables. C'est-à-dire, quelles variables binaires ont tendance à co-varier ensemble. Dans le cadre du role mining sur une matrice (utilisateurs, permissions), chaque colonne de V identifierait un ensemble de permissions qui sont typiquement accordées simultanément, révélant ainsi les rôles potentiels du système.

Vecteurs de sortie (U) : Chaque colonne de U représente un "profil type" dans l'espace des observations (lignes de la matrice). Ces vecteurs singuliers gauches caractérisent des groupes ou des motifs d'observations qui présentent des comportements similaires vis-à-vis des variables binaires. Dans le contexte d'une matrice de contrôle d'accès (utilisateurs, permissions), chaque colonne de U identifierait un type de "rôle" avec un profil d'accès spécifique aux ressources du système.

Valeurs singulières (σ_i) : Elles quantifient l'amplitude de chaque mode principal de variation. Pour une matrice binaire, une valeur singulière élevée indique qu'un motif particulier d'associations binaires est fortement présent dans les données et contribue significativement à la structure globale de la matrice.

TSVD Pour calculer la version tronquée de SVD (TSVD) qui sert à la réduction de dimension, il faut choisir un rang k. Ce rang peut aussi être appelé le nombre de composantes (components en anglais) car il indique la dimension de l'espace visé pour la décomposition. Dans ce cas, on constate que la solution est la suivante :

$$M \approxeq U\tilde{\Sigma}V^T \tag{3.2}$$

avec $\tilde{\Sigma}$ égale à Σ, si ce n'est qu'elle ne contient que les r plus grandes valeurs singulières, les autres étant remplacées par 0. C'est d'ailleurs cette propriété qui explique les qualités de réduction de bruit de cette transformation, car seuls les motifs les plus fréquents sont exprimés. Cette solution minimise la distance entre M et son approximation au sens de la norme spectrale.

Variance expliquée Lorsqu'on utilise TSVD, la variance expliquée permet de déterminer combien de composantes singulières sont nécessaires pour capturer un pourcentage donné de la variabilité totale de la matrice binaire.

La proportion de variance expliquée par la i-ème composante singulière est donnée par :

$$Var_i(M) = \frac{\sigma_i^2}{\sum_j \sigma_j^2}. \tag{3.3}$$

Où $\sum_j \sigma_j^2$ représente la variance totale de la matrice M, obtenue en sommant les carrés de toutes les valeurs singulières, notation simplifiée de $\sum_{j=1}^{r} \sigma_j^2$ avec r le rang de la matrice M. La variance cumulée expliquée par les k premières composantes est donc :

$$TVar_k(M) = \sum_{i=1}^{k} Var_i(M) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_j \sigma_j^2} \tag{3.4}$$

Cette approche permet de réduire la dimensionnalité en ne conservant que les composantes qui expliquent la majorité de la variance, facilitant ainsi l'analyse et l'interprétation des structures latentes.

Sortie On appelle par la suite réduction de la matrice M ou décomposition de la matrice M la sortie R obtenue suite au procédé TSVD avec un rang k donné. Cette matrice comporte autant de lignes que la matrice originale, mais comporte k colonnes à valeurs réelles non nulles. Mathématiquement, c'est :

$$R = U\tilde{\Sigma} \tag{3.5}$$

Dans la cadre du role mining avec une matrice UPA, la matrice R indique sur chaque ligne la représentation réduite à valeurs réelles sur les composantes des permissions d'un utilisateur. Plus les coordonnées des utilisateurs dans cet espace réduit sont proches, plus leurs permissions sont similaires.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN [63] est un algorithme de partitionnement de données aussi appelé algorithme de clustering. Il utilise essentiellement deux paramètres : une distance ϵ et un nombre de points minimum $MinPts$. Pour être considérés comme un cluster, un ensemble d'au moins $MinPts$ points doivent se trouver dans un rayon ϵ pour former un centre. Les paramètres d'entrée sont donc une estimation de la densité $MinPts/\epsilon$ de points voulue pour former les clusters. L'algorithme procède ensuite par expansion itérative en explorant l'ϵ-voisinage de chaque nouveau point identifié, permettant ainsi de délimiter progressivement l'ensemble complet des points constituant le cluster. Lorsque deux clusters communiquent par un point frontière, les clusters sont fusionnés. Cette approche par propagation locale garantit la découverte de clusters de formes arbitraires tout en identifiant automatiquement les points aberrants qui ne satisfont pas les critères de densité requis [64]. Une illustration de l'algorithme est donnée sur la figure 3.1.

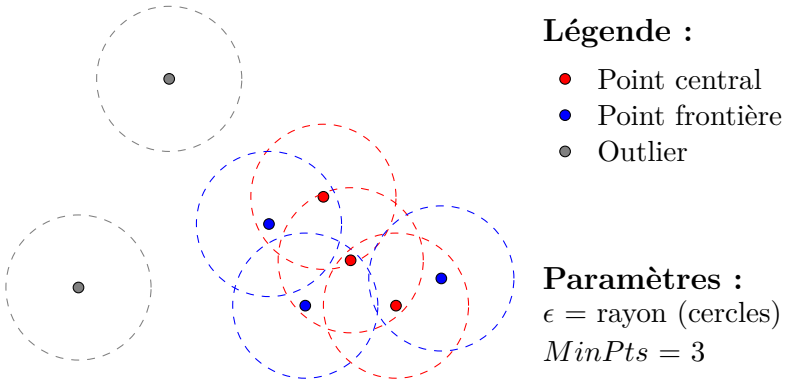


Figure 3.1 Illustration de l'algorithme DBSCAN avec un cluster identifié

Sortie La sortie de l'algorithme DBSCAN est une table à deux colonnes, la première renseignant l'identifiant d'un point observé et la seconde indiquant à quel cluster le point appartient. Les identifiants de clusters à valeurs d'entiers naturels correspondent à des clusters, et la valeur -1 correspond à l'identifiant d'un outlier.

3.1.2 Prétraitement des données

Beaucoup de jeux de données de role mining, y compris les jeux de données réels de la littérature (voir section 2.3.1) sont sous forme de table relationnelle. Une table relationnelle est un format de données ou chaque colonne désigne un attribut donné et chaque ligne représente une entrée valide dans le jeu de données. Les tables relationnelles rencontrées en role mining comportent au minimum deux colonnes : identifiant d'utilisateur et identifiant

de permission. Afin d'appliquer l'approche proposée, on a besoin d'une matrice binaire d'assignation des permissions. On utilise alors la transformation pivot.

La transformation pivot réorganise les données en changeant leur orientation : elle fait passer l'information contenue dans les valeurs des lignes vers les en-têtes de colonnes.

La transformation se divise en 3 étapes :

1. Identification des éléments dans la table relationnelle (utilisateur, permission), on identifie trois composants :
   - L'axe fixe : les utilisateurs qui deviendront les lignes de la matrice
   - L'axe pivot : les permissions qui deviendront les colonnes de la matrice
   - Les valeurs : la relation d'existence (présence/absence) qui remplira les cellules
2. Restructuration spatiale : Chaque valeur unique de la colonne "permission" devient une nouvelle colonne, et chaque valeur unique de la colonne "utilisateur" devient une ligne distincte dans la matrice UPA.
3. Remplissage : Pour chaque entrée (utilisateur, permission) valide dans la table relationnelle, on fixe l'intersection (utilisateur, permission) à 1 dans la matrice UPA et 0 si l'entrée n'existe pas.

Pour éviter toute confusion, nous utiliserons l'écriture UPA pour désigner la structure de données obtenue après prétraitement sur un jeu de données quelconque, par opposition au concept plus général de matrice UPA.

Une illustration de la transformation est renseignée sur la figure 3.2.

utilisateur	permission
123	A
123	B
123	D
456	B
456	C
789	D
789	E
789	C

$\Rightarrow$

	A	B	C	D	E
123	1	1	0	1	0
456	0	1	1	0	0
789	0	0	1	1	1

Figure 3.2 Exemple de pivot d'une table relationnelle vers une matrice binaire UPA

3.2 Réduction de dimension

La première étape de la méthode proposée est la réduction de dimension en utilisant TSVD. Il faut d'abord déterminer la nombre de composantes à utiliser pour décomposer la matrice UPA. Molloy et al. [24] fournissent un objectif clair, trouver le nombre de composantes k qui permet d'obtenir une variance cumulée d'au moins 80%. C'est-à-dire, en reprenant l'équation 3.4 :

$$TVar_k(UPA) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_j \sigma_j^2} > 80\% \tag{3.6}$$

Cette condition seule a cependant tendance à produire un nombre écrasant de composantes k si la matrice UPA est particulièrement bruitée. Puisque ce phénomène gêne la prochaine étape de notre méthode, on rajoute une condition sur la variance expliquée de la dernière composante courante k. C'est à dire, en reprenant l'équation 3.3:

$$Var_k(UPA) = \frac{\sigma_k^2}{\sum_j \sigma_j^2} < \epsilon \tag{3.7}$$

Lorsque la condition 3.7 est vérifiée, on considère qu'ajouter plus de composantes entraine des rendements décroissants. Empiriquement, $\epsilon = 0.02$ donne des résultats fiables. On utilise donc la valeur de k obtenue lorsque la condition 3.6 OU 3.7 est vérifiée.

On obtient alors une réduction de rang k de UPA selon l'équation 3.5 qu'on note R_k.

3.3 Clustering et identification des outliers

Dans le cadre du role mining, l'approche proposée utilise DBSCAN pour identifier les utilisateurs anormaux, ayant des motifs de permissions inhabituels, et de regrouper les utilisateurs aux permissions similaires. On émet en effet l'hypothèse que les instances de privilege creep se traduisent dans R_k, de sorte que les utilisateurs touchés se retrouvent éloignés des autres en raison des permissions anormales qui leur sont assignées. Reste un problème : déterminer les paramètres optimaux pour DBSCAN de sorte que l'algorithme ne fasse par rentrer d'instances de privilege creep dans des clusters d'utilisateurs.

Parmi ces paramètres, $MinPts$ correspond au nombre minimal de points (utilisateurs) pour former un cluster. $MinPts$ doit donc être fixé comme la taille minimale d'une équipe possédant des permissions similaires qu'on peut rencontrer dans l'organisation. Ceci est directement lié à la structure de l'entreprise divisée en départements, sections, équipes, etc. Des données issues des ressources humaines ou des connaissances organisationnelles permettraient d'assigner une valeur appropriée pour ce paramètre en environnement réel. Empiriquement, une valeur de

$MinPts \geq 5$ donne des résultats satisfaisants.

ϵ est plus difficile à déterminer efficacement. Pour cela, on opte pour une méthode algorithmique. On doit calculer la distance euclidienne du plus proche voisin (1-NN) de chaque point (utilisateur) sur R_k. On ordonne ensuite les distances obtenues dans une liste par valeurs croissantes. On considère la courbe C obtenue en prenant en abscisse les indices de la liste et en ordonnée les distances correspondantes. La valeur de ϵ est alors donnée par l'ordonnée du coude de la courbe. C'est-à-dire, l'endroit où la pente change brusquement.

L'identification du coude peut se faire de façon visuelle cependant on préfère une technique automatisée qui va choisir une valeur optimale. Cette technique s'appelle Normalized Difference Curve Technique (NDCT), et fonctionne en 3 étapes illustrées sur la figure 3.3 :

1. Normaliser les deux axes de la courbe C dans l'intervalle [0,1] et prendre comme courbe de référence l'identité sur l'intervalle [0,1].

2. Calculer la différence entre la courbe C et la référence sur [0,1].

3. Trouver la valeur de différence maximale qui se trouve normalement au coude.

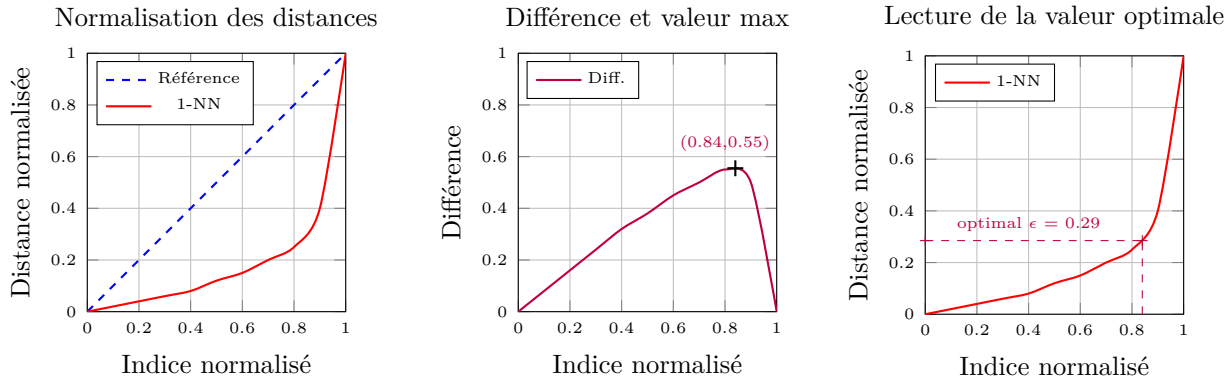


Figure 3.3 Exemple d'utilisation de la technique NDCT

Une fois le ϵ optimal déterminé, on procède au clustering des utilisateurs en utilisant leurs coordonnées sur R_k. On obtient alors une table qui renseigne pour chaque utilisateur, le cluster auquel il appartient, qu'on appelle par la suite la table utilisateur-cluster (UCT).

3.4 Nettoyage

Afin de nettoyer la matrice UPA, on utilise conjointement UPA et UCT. L'objectif est d'effectuer une analyse statistique sur chaque cluster pour déterminer quelles assignations de permissions doivent être exprimées et surtout minées.

La première étape du nettoyage est la suppression des outliers identifiés. On assure alors la qualité des rôles à miner, car on évite des potentielles instances de privilege creep. L'idée est que les autres utilisateurs "sains" sont suffisants pour inférer les rôles nécessaires pour structurer le contrôle d'accès. On crée donc une copie de la matrice UPA dans laquelle on retire les outliers et on sauvegarde leur configuration pour plus tard.

La deuxième étape du nettoyage consiste en l'analyse statistique des clusters. Pour cela, on calcule la prévalence de toutes les permissions dans chaque cluster. La prévalence d'une permission dans un cluster est définie comme la proportion d'utilisateurs au sein de ce cluster qui possèdent la permission en question. Par exemple dans un cluster de 20 personnes, si une permission a est assignée à 18 personnes et une autre permission b est assignée à 3 personnes, alors la prévalence de la permission a au sein du cluster est de $18/20 = 0{,}9$, tandis que la prévalence de la permission b est de 0,15. Comme discuté dans la partie 2.2.1, le bruit est généralement minoritaire face aux permissions légitimes ce qui fait que les permissions issues du bruit ont une prévalence plus faible que les permissions légitimes si l'étape de clustering est réussie. L'utilisation d'un seuil permet alors de supprimer directement ces permissions avec une prévalence trop faible pour chaque cluster.

En nommant le seuil de nettoyage (cleaning threshold) t_c : pour chaque cluster, on supprime les permissions avec une prévalence inférieure à t_c et on conserve dans la matrice UPA les permissions avec une prévalence supérieure à t_c. En reprenant notre exemple précédent avec les permissions a et b de prévalence 0,8 et 0,15 respectivement dans un cluster donné, en fixant $t_c = 0,8$, alors les 18 utilisateurs portant la permission a gardent cette permission, mais on supprime la permission b aux trois utilisateurs concernés.

La matrice obtenue après le processus de nettoyage est appelée $CleanedUPA$ ($CUPA$).

3.5 Role mining

Une fois la matrice $CUPA$ obtenue, on passe à l'étape de role mining. On réutilise des algorithmes de décomposition exacte déjà existants : FM [19] et Optimal Boolean Matrix Decomposition/RMP [65] pour la génération de rôles. On utilise ces algorithmes, car ce sont ceux qui offrent la meilleure performance en temps d'exécution pour des jeux de données de

taille importante (nombre d'utilisateurs supérieur à 100 et rôles minés supérieur à 100). En particulier, Optimal Boolean Matrix Decomposition/RMP utilise une heuristique qui accélère grandement le temps de calcul pour miner les rôles. L'utilisation d'un autre algorithme de role mining aurait considérablement rallongé les temps de calcul, compte tenu du fait que les algorithmes sélectionnés ont déjà requis 24 heures de calcul pour traiter le benchmark de données synthétiques généré.

Une fois les rôles générés, on utilise une approche d'algorithme glouton pour assigner les rôles aux utilisateurs sur la matrice $CUPA$. Pour chaque utilisateur, l'algorithme d'assignation de rôles trouve le rôle qui couvre le plus grand nombre de permissions d'un utilisateur et lui assigne. On répète ensuite cette étape autant de fois que nécessaires pour couvrir le reste des permissions d'un utilisateur. L'algorithme s'arrête une fois que l'intégralité des permissions d'un utilisateur sont couvertes. Encore une fois, c'est l'approche qui produit les résultats les plus rapides en plus d'être une approche exacte.

Une fois l'algorithme de role mining et celui d'assignation roulé, on obtient donc des matrices UP et PA incomplètes, ne contenant pas les outliers retirés lors du nettoyage.

3.6 Réunification

Si on s'arrête à l'étape de role mining, l'approche proposée est incomplète, car on n'a pas réintégré les outliers à la matrice $CUPA$. En effet, en milieu réel, on ne cherche pas à supprimer des utilisateurs du système de contrôle d'accès. Pour résoudre cette problématique, on introduit deux concepts originaux : la réunification omnisciente et la réunification heuristique, contributions spécifiques à cette recherche.

3.6.1 Réunification omnisciente

La réunification omnisciente est un processus fictif qui imagine l'existence d'un administrateur système capable de nettoyer les permissions d'un utilisateur parfaitement : retirer bruit et privilege creep en gardant les permissions légitimes. On utilise cette abstraction afin de pouvoir évaluer les performances de l'approche proposée plus facilement. En effet, lorsqu'on fait face à des jeux synthétiques, on utilise la réunification dite omnisciente pour nettoyer les permissions des outliers, en prenant leur configuration légitime avant ajout de bruit et de privilege creep lorsqu'on a généré le jeu de données. Cette limitation est prise en compte lors de l'évaluation, car on pénalise les faux positifs de détection d'anomalies dans les métriques d'évaluation.

3.6.2 Réunification heuristique

La réunification dite heuristique est l'approche utilisée dans le cadre des systèmes réels pour fournir une recommandation de réintroduction des outliers à un administrateur système. C'est le cas dans lequel l'outil proposé est utilisé dans un contexte organisationnel chez le partenaire industriel de cette recherche.

Deux approches concurrentes sont explorées :

- La première approche consiste à attribuer les rôles extraits aux outliers, même si toutes leurs permissions peuvent ne pas être couvertes, en utilisant le même algorithme glouton que pour le role mining. Cela donne une idée approximative des rôles potentiels que les outliers pourraient avoir, et fait également ressortir les permissions potentiellement problématiques qui ne sont pas couvertes.
- La deuxième approche utilise les informations sur les clusters. Plus précisément, le barycentre de chaque cluster en utilisant les coordonnées binaires (qui correspond au vecteur de prévalences). Chaque outlier est ensuite lié au barycentre le plus proche de lui. Cela fournit un aperçu sur le cluster auquel pourrait appartenir l'outlier en question.

En fin de compte, la décision de réunification doit être prise par un administrateur réel, chargé de réviser les accès des comptes utilisateurs. En effet, les deux approches pourraient fournir des informations contradictoires, qui pourraient être élucidées avec l'utilisation d'attributs par exemple.

3.7 Résumé

Mon approche de role mining sensible au privilege creep est résumée dans le diagramme 3.4. Les structures de données obtenues (matrices, tables, etc.) sont indiquées en vert et les transformations successives pour obtenir ces structures sont renseignées en rouge. Dans les boîtes hexagonales sont renseignées les informations importantes à prendre en compte lors du role mining.

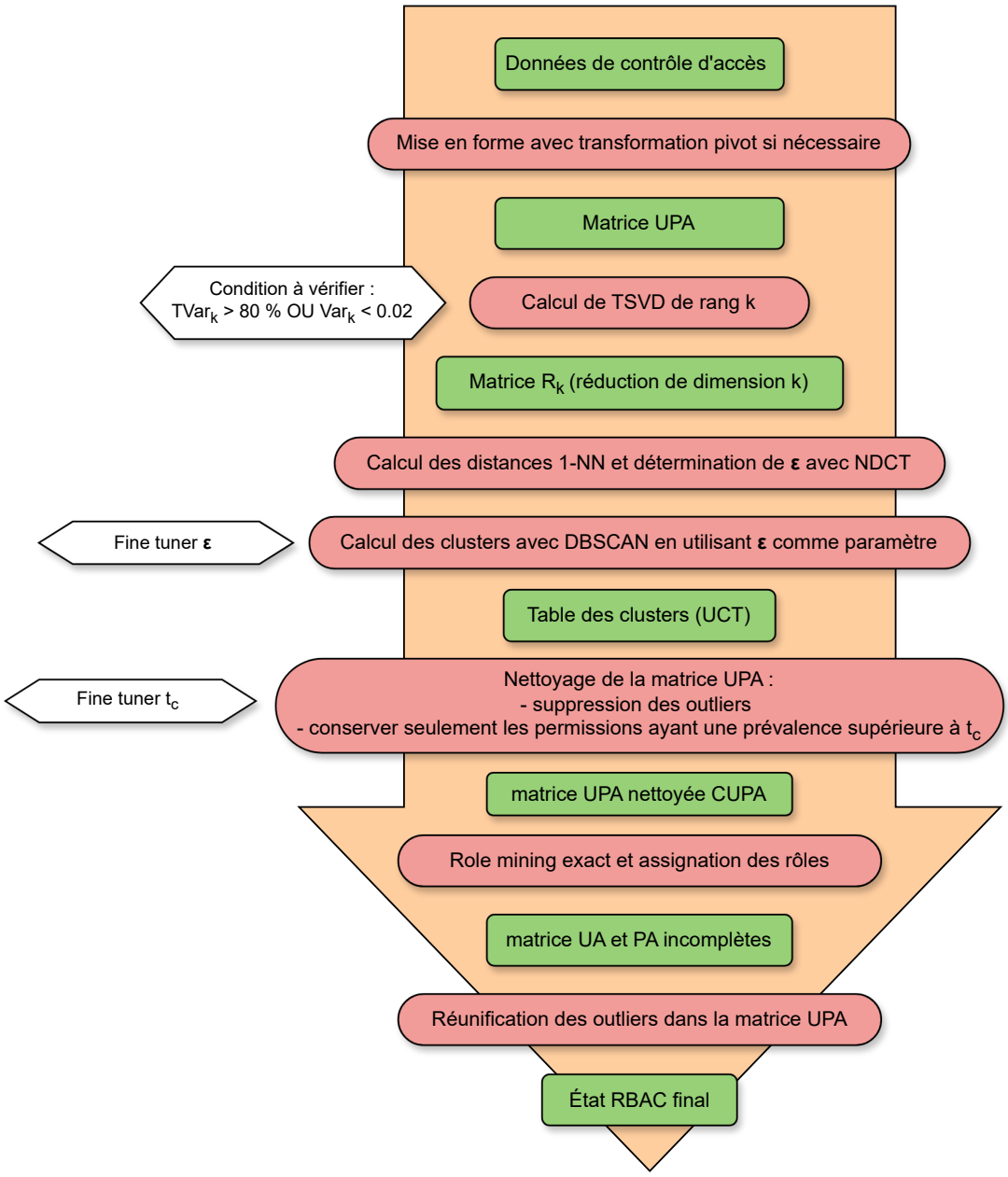


Figure 3.4 Diagramme résumant l'approche de role mining sensible au privilege creep

CHAPITRE 4 GÉNÉRATION DE JEUX DE DONNÉES SYNTHÉTIQUES

Ce chapitre présente la méthode de génération de données synthétiques utilisée pour évaluer la méthode de role mining proposée. Il est divisé en quatre parties : la première définit une nomenclature des anomalies identifiées et devant être ajoutées aux jeux de données générés, la seconde énonce les hypothèses utilisées pour construire le générateur et qui ont motivé les choix de la partie suivante, ensuite le générateur en lui-même est décrit en détail en commençant par la génération d'assignations légitimes puis l'ajout d'anomalies, enfin l'approche au complet est résumée à l'aide de diagrammes pour aider la compréhension.

4.1 Nomenclature des anomalies

On appelle anomalie toute assignation qui n'est pas une assignation légitime et/ou qui n'est pas applicable. On désigne une assignation légitime comme toute permission dont un utilisateur a besoin pour accomplir ces tâches. Pareillement, on appelle permission non applicable toute permission, légitime ou non, qu'il n'est pas nécessaire d'exprimer dans la structure de contrôle d'accès choisie : On préfère les exprimer sous forme d'exception. On rejoint la pensée exprimée par Vaidya et al. [22] selon laquelle exprimer l'entièreté des assignations complexifie grandement l'état RBAC résultant. Aussi notre approche de role mining n'est pas exacte et vise le nettoyage du jeu de données rencontré, il est aussi nécessaire d'introduire cette classe d'assignations dont l'objectif est de ne pas les exprimer.

Bien que d'autres anomalies puissent exister, on se limite à celles détaillées dans cette nomenclature. En effet, toutes les anomalies rencontrées dans la revue de littérature sont au moins incluses dans la nomenclature suivante. Il n'y a donc pas de perte de généralité vis-à-vis des recherches précédentes.

4.1.1 Bruit

Pour mieux comprendre cette nomenclature, on suppose qu'on prend une structure de contrôle d'accès RBAC.

- Bruit de correction : erreurs d'administration isolées qui surviennent généralement lorsqu'un utilisateur passe par le processus de provisionnement d'accès (provisioning) pour la première fois, n'affectant qu'un petit ensemble de permissions. Le bruit de correction est généralement additif, c'est-à-dire qu'il entraîne des attributions de permissions supplémentaires sur les utilisateurs affectés, en raison du problème de disponibilité

que cause le bruit soustractif. Ces erreurs d'administration touchent principalement des permissions qui sont légitimes pour d'autres utilisateurs, car on a surévalué le "need to know" d'un utilisateur. D'autres raisons peuvent être trouvées pour ce type d'erreur, mais ce qu'il faut retenir est le caractère isolé de ces permissions.

- Bruit d'applicabilité RBAC : permissions légitimes qui ne sont pas suffisamment partagées entre les utilisateurs pour être utilement exprimées dans le modèle RBAC. En effet, les exprimer augmenterait la complexité de l'état RBAC et compromettrait donc les avantages de gestion liés au maintien de cette structure pour le contrôle d'accès. On peut donner comme exemple de permissions non applicable à RBAC les permissions quasi discrétionnaires d'accès aux dossiers personnels, puisque seuls la personne concernée par ces permissions ainsi que l'administrateur du système y ont accès. Ce sont typiquement ce genre de permissions qu'il faut considérer comme des exceptions. Une règle empirique en role mining est la règle des 80-20 : 80% des assignations doivent être couvertes en utilisant le modèle RBAC tandis que 20% peuvent être considérées comme des exceptions.

Dans notre nomenclature, on fait le choix de désigner le bruit de correction comme un bruit purement additif. En effet, le bruit soustractif ne pose pas un problème inhérent de sécurité, mais un problème de disponibilité qui est généralement vite identifié et corrigé en conditions réelles.

4.1.2 Accumulation de privilèges (privilege creep)

Ici, on désigne par le terme acteur de la menace interne une personne ayant des identifiants valables dans le système de GIA d'une entreprise, comme un employé. Deux scénarios de privilege creep sont alors considérés :

- Scénario 1 : un acteur de la menace interne change de poste au sein de l'organisation, mais conserve un sous-ensemble de permissions assignées pour ses fonctions précédentes en raison d'un déprovisionnement incomplet. Un autre exemple est lorsqu'un employé démissionne, et qu'un autre employé se voit accorder les permissions additionnelles pour accomplir les tâches de l'ancien employé, permissions qui ne seront pas ou partiellement révoquées une fois un remplaçant trouvé. Appelons cela privilege creep de type I.
- Scénario 2 : un groupe d'acteurs de la menace interne s'est vu assigné à un projet temporaire désormais terminé. Des permissions supplémentaires leur ont été accordées pour accomplir leurs nouvelles responsabilités au sein de ce projet. Cependant, ces

assignations supplémentaires n'ont pas été entièrement révoquées à la fin du projet. Cela peut se produire en raison de registres de projets pas ou mal maintenus. Appelons cela privilege creep de type II.

La différence clé entre ces deux types de dérive des privilèges est le sous-ensemble de permissions impactées.

- Pour le privilege creep de type I : Les permissions impactées sont encore légitimes pour d'autres employés. En effet, ceux qui occupent actuellement un poste similaire au poste précédemment détenu par l'employé ayant subi une dérive des privilèges, ou l'employé qui prend les responsabilités du poste vacant, ont un besoin valide de ces permissions pour accomplir leur tâche.
- Pour le privilege creep de type II : Les permissions impactées ne sont pas légitimes pour les employés extérieurs au projet. En effet, les permissions sont liées au projet et une fois le projet terminé, aucun employé ne devrait les conserver.

D'autres scénarios peuvent se traduire par une combinaison de ces deux types de privilege creep. Par exemple, lors de la fusion de deux entreprises pendant une acquisition, la période d'intégration peut générer simultanément les deux types :

- Type I : Un employé de l'entreprise acquise conserve des privilèges administratifs de son ancien rôle (par exemple, administration du système RH), alors que ces responsabilités ont été transférées à quelqu'un d'autre. Ces privilèges restent légitimes et nécessaires pour le nouvel administrateur.
- Type II : Le même employé garde également des accès aux outils temporaires créés spécifiquement pour la migration (serveurs de test, plateformes de synchronisation des données), qui ne sont plus nécessaires pour personne une fois l'intégration terminée.

Cette situation hybride est particulièrement problématique, car elle cumule les risques sur un même individu.

Il existe aussi probablement d'autres types d'instances de privilege creep reliées à d'autres scénarios, mais on ne considère que les scénarios décrits ici pour la génération de données synthétiques.

4.2 Hypothèses sur la génération des jeux de données

La génération d'une matrice UPA (voir 2.1.1) est l'objectif visé de la méthode de génération de données synthétiques proposée. Les attributs des utilisateurs ne sont pas générés, car on n'a tout simplement pas de connaissances préalables à utiliser pour les modéliser de manière réaliste.

L'approche proposée se concentre sur les assignations de permissions persistantes, c'est-à-dire à durée indéterminée, ignorant les privilèges Just-In-Time (JIT) et toute autre technologie qui rend les assignations de permissions dynamiques vis-à-vis de certains paramètres (heure, lieu, etc.). Les privilèges JIT sont en effet utilisés sur des plages de temps extrêmement courtes, allant de quelques minutes à quelques heures et ne dépassant pas 48h, et expirent automatiquement. On ne considère donc que des configurations statiques : soit une représentation figée d'un système à un moment donné, soit une traduction statique d'un système dynamique en prenant les permissions maximales accordées à chaque utilisateur sur une période donnée. Cette approche permet de capturer l'ensemble des droits d'accès potentiels sans tenir compte de leur variabilité temporelle.

On ne considère pas l'existence de comptes dormants, qui est un autre problème que celui traité dans cette recherche et qui relève de la gestion lorsqu'un acteur de la menace interne quitte l'organisation. De plus pour identifier un compte comme dormant, des attributs sur les utilisateurs sont généralement nécessaires comme le temps de dernière connexion, les informations de ressources humaines sur un employé, etc.

4.3 Générateur de jeux de données synthétiques

Le générateur de données synthétiques développé dans cette recherche prend inspiration sur le générateur de données "Tree" de Molloy et al. [33]. Les principales modifications portent sur la stratégie de propagation des nœuds utilisés pour générer les arbres, le processus d'ajout de bruit, et le nouveau processus d'ajout d'instances de privilege creep. Une attention particulière est portée sur la flexibilité du générateur de données, d'où l'introduction de nombreux paramètres servant à créer des jeux de données diversifiés.

Cette section est organisée en trois parties qui traitent respectivement la génération des permissions légitimes avec la structure d'arbre, l'ajout de bruit, et finalement l'ajout d'instances de privilege creep.

4.3.1 Générer les permissions légitimes

Pour générer les permissions légitimes, on propose d'utiliser une structure arborescente qui mime une hiérarchie d'entreprise fictive. Chaque nœud représente une subdivision de l'entreprise sur laquelle on assigne des permissions, et les utilisateurs sont assignés sur les feuilles de cette structure arborescente. Le nœud racine représente donc les permissions à l'échelle de "l'organisation", que tous les utilisateurs hériteront. Chaque nœud après le nœud racine représente une subdivision de l'organisation : vice-présidence, départements, équipes et ainsi de suite. Finalement, les nœuds feuilles représentent la plus petite subdivision de l'organisation fictive. Les utilisateurs héritent alors des permissions assignées aux nœuds aux dessus d'eux en cascade. On désigne par la suite cette structure arborescente avec des permissions et utilisateurs sous le nom de modèle.

En tant qu'objectif de RM, les rôles minés doivent correspondre à la structure du modèle. Sachant cela, les modèles sont utilisés pour créer des hiérarchies de permissions légitimes. Le générateur proposé commence par le nœud racine, puis fait croître l'arbre en propageant les nœuds enfants de manière itérative. On utilise six paramètres pour construire les arbres :

- *min_depth* et *max_depth* à valeurs dans $\mathbb{N}^*$ avec $min\_depth \leq max\_depth$, encodent respectivement la profondeur minimale et maximale de l'arbre. Entre ces bornes, la probabilité qu'un nœud cesse de se propager augmente linéairement avec la profondeur. C'est-à-dire que jusqu'à *min_depth*, tous les nœuds continuent obligatoirement à se propager (probabilité = 1). À partir de *min_depth*, la probabilité de propagation diminue linéairement jusqu'à atteindre 0 à *max_depth*. La probabilité de propagation à la profondeur d, $p(d)$ est alors définie comme suit :

$$p(d) = \begin{cases} 1 & \text{si } d \leq min\_depth \\ \dfrac{max\_depth - d}{max\_depth - min\_depth} & \text{si } min\_depth < d < max\_depth \\ 0 & \text{si } d \geq max\_depth \end{cases} \tag{4.1}$$

  À chaque tentative de propagation d'un nœud situé à une profondeur d, un tirage aléatoire détermine si le nœud continue à se propager. En implémentation, un nombre aléatoire u est tiré uniformément dans l'intervalle $[0, 1]$. Le nœud se propage si et seulement si $u \leq p(d)$.

- *min_children* et *max_children* à valeurs dans $\mathbb{N}^*$ avec $min\_children \leq max\_children$, encodent respectivement le nombre minimum et maximum de nœuds enfants par nœud.

- *avg_branch* et *std_dev* à valeurs dans $\mathbb{R}^*$, respectivement le facteur d'embranchement et l'écart-type sur ce facteur d'embranchement, encodent comment les nœuds se propagent dans l'arbre. Le nombre de nœuds enfants suit une distribution normale centrée autour de *avg_branch* avec un écart-type de *std_dev*, ramené à des entiers. Avec une notation probabiliste, le nombre de nœuds enfants $N_{enfants}$ générés pour chaque nœud suit :

$$N_{enfants} \sim \lfloor \mathcal{N}(avg\_branch, std\_dev^2) \rfloor \tag{4.2}$$

  où $\mathcal{N}(\mu, \sigma^2)$ représente une distribution normale de moyenne μ et de variance σ^2, et $\lfloor \cdot \rfloor$ désigne la fonction partie entière. Les fonction min et max sont ensuite utilisées pour borner le nombre obtenu entre *min_children* et *max_children* :

$$N_{enfants} \leftarrow \max(min\_children, \min(max\_children, N_{enfants})) \tag{4.3}$$

Ainsi, la propagation des nœuds enfants est probabiliste non uniforme. Aussi les arbres produits peuvent ne pas être équilibrés, puisque la profondeur de l'arbre est, elle aussi, soumise à un processus probabiliste. Avec ces paramètres, plusieurs structures d'arbres différentes peuvent être créées. Par exemple, des arbres plats qui ne sont pas très profonds (grand facteur d'embranchement et une profondeur faible) ou plutôt des arbres profonds avec peu de branches (faible facteur d'embranchement et grande profondeur).

Un exemple de structure arborescente obtenue est renseignée sur la figure 4.1.

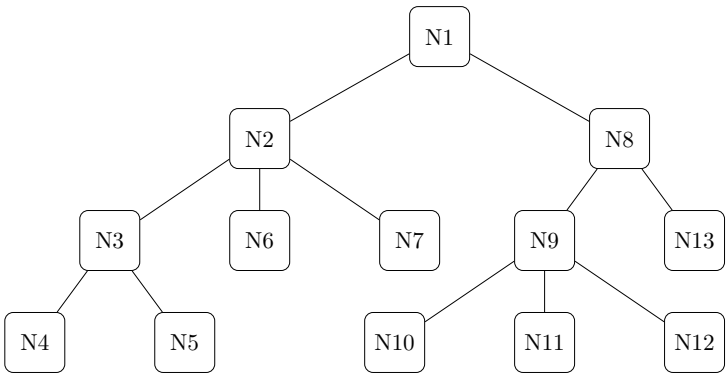


Figure 4.1 Exemple de structure d'arbre avec numérotation itérative des nœuds

Les utilisateurs et les permissions sont ensuite ajoutées de façon probabiliste sur les nœuds. On utilise alors quatre paramètres :

- *min_user* et *max_user* à valeurs dans $\mathbb{N}^*$ avec $min\_user \leq max\_user$, qui encodent respectivement le nombre minimal et maximal d'utilisateurs sur les feuilles.
- *min_perm* et *max_perm* à valeurs dans $\mathbb{N}^*$ avec $min\_perm \leq max\_perm$, qui encodent respectivement le nombre minimal et maximal de permissions sur chaque nœud.

On assigne ensuite les permissions et les utilisateurs à l'aide d'une loi binomiale à valeurs entre les extremums. Le nombre de permissions N_{perm} attribué aux nœuds suit alors la loi :

$$N_{perm} \sim min\_perm + \mathcal{B}(max\_perm - min\_perm, 0.5) \tag{4.4}$$

Où $\mathcal{B}(n, p)$ représente une loi binomiale avec n essais et probabilité de succès $p = 0.5$. Cette formulation garantit que le nombre de permissions N_{perm} est compris entre *min_perm* et *max_perm*. La probabilité $p = 0.5$ assure une distribution symétrique, permettant une répartition équilibrée, mais aléatoire des permissions sur les nœuds de l'arbre.

De même, on a pour les assignations d'utilisateurs sur les feuilles, la formule :

$$N_{user} \sim min\_user + \mathcal{B}(max\_user - min\_user, 0.5) \tag{4.5}$$

Ces choix mathématiques sont motivés par un argument statistique : Dans les organisations réelles, le nombre de permissions par rôle ou d'utilisateurs par équipe résulte de multiples facteurs indépendants (besoins métier, contraintes réglementaires, structure hiérarchique, ressources disponibles, etc.). Qualitativement, cette multiplicité de facteurs produit un comportement statistique caractéristique : la plupart des rôles ont un nombre "moyen" de permissions (correspondant aux besoins standard), tandis que les cas extrêmes (très peu ou beaucoup de permissions) sont naturellement plus rares. Il en va de même pour la taille des équipes au sein d'une organisation. Une modélisation avec une loi binomiale, approchant une loi normale, semble donc plus appropriée qu'une distribution uniforme.

Une fois les permissions et utilisateurs assignés, on obtient alors un modèle de permissions légitimes. Un exemple est renseigné sur la figure 4.2.

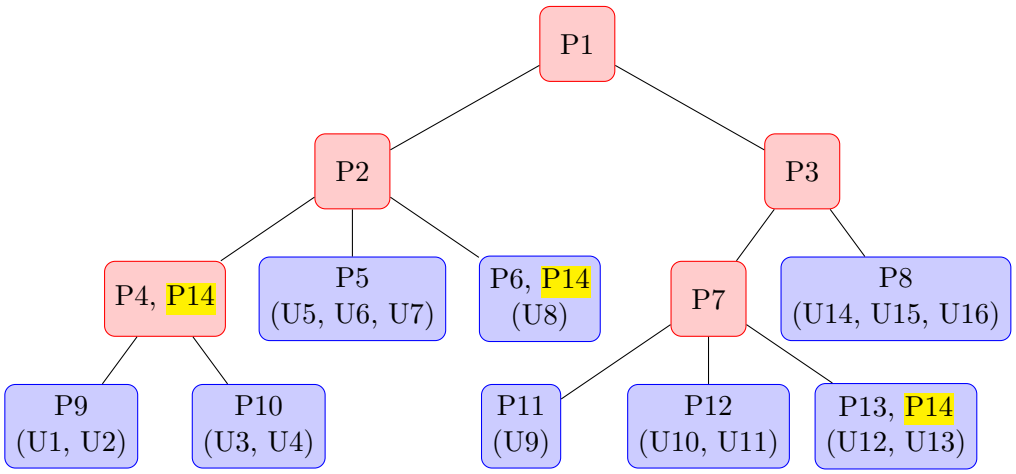


Figure 4.2 Exemple d'arbre de permissions légitimes générés

En rouge les nœuds portant des ensembles de permissions et en bleu des feuilles portant des ensembles de permissions et des utilisateurs. En surligné, les ensembles de permissions transversales

Une dernière étape consiste à enrichir le jeux de données avec des ensembles de permissions dites "transversales". Ce sont des ensembles de permissions légitimes qui sont partagés sur différents nœuds de la hiérarchie, mimant par exemple des applications ou configurations partagées entre plusieurs départements de l'organisation. On rajoute donc des ensembles de permissions sur l'arbre généré entre plusieurs nœuds pris au hasard. Par exemple sur la figure 4.2 on a rajouté un ensemble de permission $P14$.

On fixe arbitrairement le nombre d'ensembles de permissions transversales à 2 ou 3. Ce choix vise à éviter une croissance excessive du nombre total de permissions lorsque les paramètres limitent fortement le nombre de permissions par nœud. En effet, un nombre plus élevé d'ensembles transversaux entraînerait une multiplication importante des permissions, même avec des contraintes restrictives au niveau local. Le nombre de permissions contenu dans ces ensembles de permissions transversaux suit la même loi utilisée pour générer les ensembles sur les nœuds, et le nombre de nœuds reliés ensemble à 2 ou 3, là encore pour les mêmes raisons que citées précédemment.

Une fois ce processus achevé, on traduit ce modèle en matrice UPA avec les règles d'héritage de permission énoncées précédemment. On appelle par la suite la matrice générée à cette étape la matrice UPA légitime.

4.3.2 Ajouter le privilege creep

Comme identifié dans la section 4.1.2, deux types de privilege creep sont ajoutés à la matrice UPA légitime : le privilege creep de type I et celui de type II. Afin de contrôler les quantités de ces instances, on introduit trois paramètres :

1. p_I la proportion d'utilisateurs qui présentent des cas de PC de type I. Par exemple, si $p_I = 0.05$, alors 5% des utilisateurs vont être victime de privilege creep de type I.

2. c_I la proportion d'assignations de permissions copiées d'un autre utilisateur dans une instance de PC de type I. Par exemple, si $c_I = 0.2$, alors 20% des permissions d'un utilisateur sont utilisées pour créer une instance de PC chez un autre.

3. q_{II} la quantité de permissions ajoutées pour les utilisateurs qui présentent une PC de type II. Tous les utilisateurs affectés se voient alors accorder ces permissions. Par exemple, si $q_{II} = 50$, alors 50 permissions supplémentaires sont ajoutées à la matrice UPA et assignées aux utilisateurs avec PC dans le même projet.

Afin d'avoir toujours un ensemble varié d'instances de PC, les règles suivantes sont utilisées :

- 30% des instances de PC de type I utilisent $c_I = 1$. On appelle ces instances privilege creep de type I total.

- 70% des instances de PC de type I se voient attribuer des valeurs de c_I décroissantes linéairement de 1 à 0. Par exemple, si 5 utilisateurs sont dans ce cas, la plage de valeurs assignées pour c_I est 1, 0.8, 0.6, 0.4 et 0.2. On appelle ces instances, privilege creep de type I partiel.

- Le nombre d'utilisateurs dans chaque instance de PC de type II est fixé à 8, et le nombre d'instances $n_{\text{instances Type II}}$ est calculé en utilisant la formule suivante :

$$n_{\text{instances Type II}} = \lfloor \log_{10}(n_{\text{utilisateurs}}) \rfloor \tag{4.6}$$

  Cette formule permet d'obtenir un nombre de projets temporaires qui croit de façon logarithmique en le nombre total d'utilisateurs. En effet, empiriquement une croissance linéaire a été testée, mais elle produisait trop d'instances pour les grands jeux de données. Ainsi donc, plusieurs groupements d'utilisateurs sont créés au hasard et on leur assigne un nombre q_{II} de permissions additionnelles.

Grâce à ces paramètres, il est possible de produire des instances de privilege creep de fréquence et de magnitude variées. On injecte alors les assignations supplémentaires créées par ce processus dans la matrice UPA légitime. On appelle la matrice à cette étape la matrice UPA avec privilege creep (privilege crept UPA).

Détails d'implémentation

Lors de l'injection d'une instance de privilege creep de type I, on sélectionne aléatoirement deux utilisateurs a et b issus de deux équipes distinctes, afin d'assurer que leurs ensembles de permissions initiaux diffèrent. Cette anomalie émule le scénario suivant : l'utilisateur a, qui appartient actuellement à l'équipe 1, faisait auparavant partie de l'équipe 2 où il détenait les mêmes privilèges que l'utilisateur b. Lors de son changement d'équipe, a a conservé ses anciens privilèges en plus de ceux de sa nouvelle équipe.

Concrètement, l'utilisateur a est celui qui est affecté par le privilege creep, tandis que l'utilisateur b représente son profil d'autorisations passé. On sélectionne l'ensemble des permissions assignées à b, puis, en fonction du paramètre c_I pour cette instance, on en extrait un sous-ensemble aléatoire. Ce sous-ensemble de permissions est ensuite assigné directement à a, s'ajoutant à ses permissions actuelles. Le tirage des utilisateurs se fait avec remise.

Pour l'injection d'une instance de privilege creep de type II, on sélectionne 8 utilisateurs distincts au hasard. On vient ensuite leur donner q_{II} permissions additionnelles. Tous les utilisateurs dans l'instance héritent du même ensemble de permissions.

Pour l'injection d'une instance de privilege creep de type II, on sélectionne aléatoirement 8 utilisateurs distincts. On assigne ensuite q_{II} permissions additionnelles à chacun des 8 utilisateurs sélectionnés. Cette configuration émule un projet temporaire terminé dont les participants ont conservé des permissions qui auraient dû être révoquées. Tous les utilisateurs de l'instance reçoivent le même ensemble de permissions, reflétant leur participation commune au projet temporaire.

On garde aussi une liste des utilisateurs touchés par le privilege creep afin d'avoir une information de référence pour les résultats (voir chapitre 5). Les utilisateurs touchés par plusieurs instances de privilege creep n'apparaissent qu'une fois dans cette liste, mais sont décomptés individuellement dans le détail des instances.

Par exemple, avec 3 utilisateurs a, b et c :

- Instance 1 (Type I) : a reçoit des permissions de b.
- Instance 2 (Type I) : c reçoit des permissions de a.
- Instance 3 (Type II) : a accumule des permissions supplémentaires.

Dans ce cas, la liste des utilisateurs anormaux est $[a, c]$, mais le décompte par instance montrerait 2 instances de type I et 1 instance de type II, pour un total de 3 instances de privilege creep.

4.3.3 Ajouter le bruit

Pour rendre le jeu de données plus réaliste, du bruit est ajouté. Ceci suit la discussion de la section 4.1.1 sur l'identification du bruit. Le bruit d'applicabilité RBAC se manifeste par des permissions additionnelles sans impact sur les permissions légitimes tandis que le bruit de correction touche seulement les permissions légitimes. Le bruit d'applicabilité RBAC est ajouté en utilisant deux paramètres de quantité et de densité, et le bruit de correction est ajouté en utilisant un paramètre de pourcentage :

- p_{bruit} : le pourcentage de bruit désigne le ratio d'assignations de permissions issues du bruit à ajouter à la matrice UPA avec privilege creep, exprimé comme une proportion du nombre d'assignations légitimes. Supposons que le pourcentage de bruit soit fixé à 30% et que la matrice UPA légitime ait 200 assignations, alors 60 assignations supplémentaires doivent être ajoutées.

- d_{bruit} : la densité de bruit désigne à quel point les assignations de permissions bruitées ajoutées sont denses. Le nombre de permissions distinctes à ajouter $N_{\text{perm bruit}}$ doit être calculé pour correspondre au paramètre de densité de bruit. On exprime cette contrainte avec une égalité du nombre d'assignations bruitées à ajouter :

$$n_{\text{legit}} \cdot p_{\text{bruit}} = N_{\text{perm bruit}} \cdot N_{\text{utilisateurs}} \cdot d_{\text{bruit}} \tag{4.7}$$

  Où n_{legit} est le nombre d'assignations légitimes et $N_{\text{utilisateurs}}$ le nombre total d'utilisateurs. À gauche de l'égalité, on a le nombre d'assignations calculées avec la définition de p_{bruit}, et à droite de l'égalité le nombre d'assignations calculées avec la définition de d_{bruit}. En remaniant l'égalité 4.7, on obtient la formule suivante :

$$N_{\text{perm bruit}} = \frac{N_{\text{legit}} \cdot p_{\text{bruit}}}{d_{\text{bruit}} \cdot N_{\text{utilisateurs}}} \tag{4.8}$$

- $p_{\text{legit-bruit}}$: le pourcentage de bruit sur les permissions légitimes indique le ratio d'assignations supplémentaires à introduire relativement au nombre de permissions issues du bruit déjà injectées. Le nombre d'assignations supplémentaires induites par le processus de bruitage des permissions légitimes est calculé par :

$$n_{\text{legit-bruit}} = n_{\text{legit}} \cdot p_{\text{bruit}} \cdot p_{\text{legit-bruit}} \tag{4.9}$$

  où n_{legit} est le nombre d'assignations légitimes. Ces assignations sont purement additionnelles, ne faisant que transformer les 0 en 1 dans la matrice UPA avec privilege creep. Supposons que ce paramètre soit fixé à 10% en utilisant le même exemple qu'avant : avec 200 assignations légitimes et 30% de bruit, on ajoute $200 \times 0.1 \times 0.3 = 6$ assignations supplémentaires sur les permissions légitimes.

On nomme la matrice obtenue après l'étape de bruitage la matrice UPA bruitée (noised UPA).

Détails d'implémentation

Les assignations de bruit d'applicabilité RBAC sont générées avec une expérience de Bernoulli sur une matrice de taille $(n_{\text{utilisateurs}}, N_{\text{bruit}})$ avec $p = d_{\text{noise}}$. Essentiellement, on a une probabilité p qu'une permission donnée soit assignée à un utilisateur donné. Une fois cette matrice binaire générée, elle est concaténée à la matrice UPA contenant le privilege creep.

Pour ce qui est du bruit de correction avec des assignations supplémentaires faites sur les permissions légitimes, on sélectionne au hasard une permission légitime et un utilisateur. Si l'assignation entre cet utilisateur et cette permission n'existe pas, on assigne cette permission à cet utilisateur. Sinon, on sélectionne à nouveau un utilisateur et une permission légitime au hasard. En répétant ce processus autant de fois que nécessaire pour assigner $n_{legit-bruit}$ permissions supplémentaires, on génère le bruit de correction voulu.

4.4 Résumé

On renseigne dans le tableau 4.1 un résumé des anomalies de bruit et de privilege creep ajoutées lors du processus de génération de jeux de données avec le détail de leur méthode de génération.

Type d'anomalie	**Permissions touchées**	**Méthode de génération**
Privilege Creep Type I - Total	Permissions légitimes d'un autre utilisateur	Sélection aléatoire de deux utilisateurs dans des équipes distinctes et copie complète des permissions de l'utilisateur source vers l'utilisateur cible. Représente 30% des instances de Type I.
Privilege Creep Type I - Partiel	Permissions légitimes d'un autre utilisateur	Sélection aléatoire de deux utilisateurs dans des équipes distinctes et copie partielle (paramétrée par $c_I \neq 1$) des permissions de l'utilisateur source vers l'utilisateur cible. Représente 70% des instances de Type I.
Privilege Creep Type II	Permissions additionnelles communes à un groupe	Sélection de 8 utilisateurs distincts au hasard. Assignation de q_{II} permissions additionnelles identiques à tous les utilisateurs du groupe. Nombre d'instances : $\lfloor \log_{10}(n_{\text{utilisateurs}}) \rfloor$.
Bruit d'applicabilité RBAC	Permissions additionnelles bruitées (distinctes des légitimes)	Expérience de Bernoulli sur une matrice de taille $(n_{\text{utilisateurs}}, N_{\text{perm bruit}})$ avec probabilité $p = d_{\text{bruit}}$. La matrice générée est concaténée à la matrice UPA avec privilege creep.
Bruit de correction	Permissions légitimes existantes bruitées	Sélection aléatoire répétée d'un utilisateur et d'une permission légitime jusqu'à générer $n_{\text{legit-bruit}}$ assignations supplémentaires.

Table 4.1 Récapitulatif des anomalies introduites dans la génération de jeux de données

Le processus entier de génération de jeux de données synthétiques est résumé sur le diagramme 4.3. Les structures de données obtenues sont indiquées en vert et les opérations successives pour obtenir ces structures sont renseignées en rouge. On renseigne aussi les 16 paramètres utilisés durant le processus de génération dans les encadrés hexagonaux.

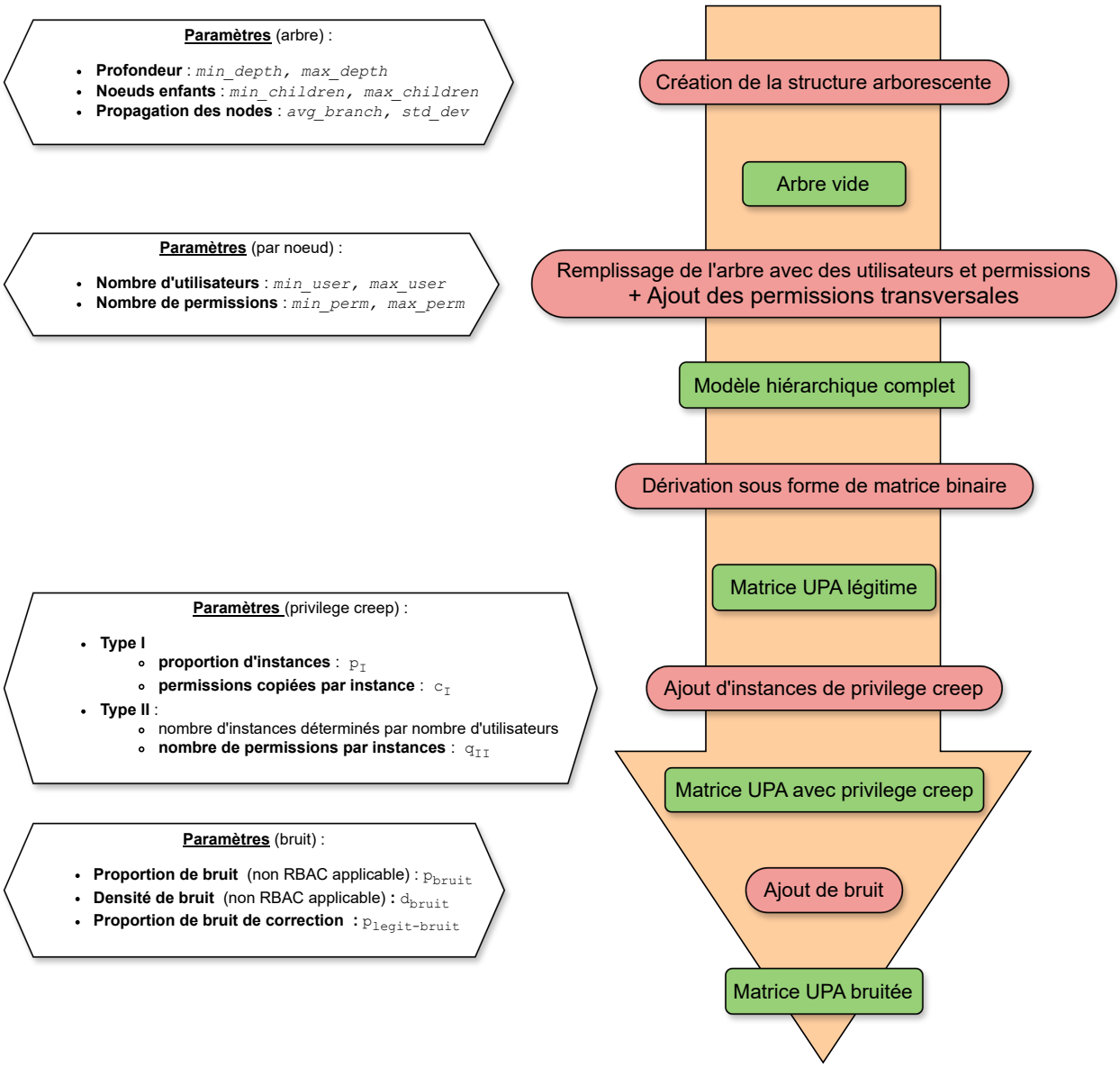


Figure 4.3 Diagramme résumant le processus de génération des jeux de données

CHAPITRE 5 ÉVALUATION

Le chapitre détaille le cadre de l'évaluation de la méthode de role mining proposée. Il commence par la présentation des benchmarks de jeux de données synthétiques utilisés pour l'évaluation, en détaillant les profils utilisés pour générer les jeux de données. Viennent ensuite la présentation des jeux de données réels, incluant les jeux publics issus de la littérature et les jeux du partenaire industriel provenant d'Active Directory, transformés en matrices UPA pour l'évaluation. Le chapitre se termine par la description des métriques d'évaluation.

5.1 Benchmark de jeux de données synthétiques

Pour les jeux de données synthétiques, on procède à la création de deux benchmarks : l'un avec des niveaux variables de bruit et de privilege creep, et l'autre avec des niveaux variables de "tension", c'est-à-dire de limitation du nombre d'utilisateurs ou de permissions.

Afin d'avoir un benchmark complet et diversifié, on crée plusieurs profils pour la génération de la structure de permissions légitimes utilisés. La table 5.1 définit les profils utilisés pour construire les arbres modèles (voir section 4.3.1), sélectionnés pour avoir une variété de profondeur et de ramification, imitant différentes structures organisationnelles.

Nom	**children**		**depth**		**avg_branch**	**std_dev**
	min	**max**	**min**	**max**		
large_flat	2	4	2	4	3	1.5
small_flat	1	5	1	3	3	1
large_string	1	2	10	15	1.7	0.5
small_string	1	2	5	8	1.6	0.4
binary_tree	1	3	2	5	2	0
highly_random	1	6	2	5	2	2

Table 5.1 Profils pour générer les arbres modèles de permissions légitimes

5.1.1 Niveaux variables de bruit et de privilege creep

Pour la première passe de génération de jeux de données, on propose de faire une évaluation des performances de l'approche pour différents niveaux de bruit et de privilege creep. On se restreint ici à cinq profils différents :

- un profil témoin sans bruit (no noise no pc)
- un profil avec une quantité et une densité de bruit faible (low noise, low density)
- un profil avec une quantité importante de bruit et une densité faible (high noise, low density)
- un profil avec une quantité faible de bruit et une densité élevée (low noise, high density)
- un profil avec une quantité de bruit et une densité élevée (high noise, high density)

Une des hypothèses formulées lors de la création de ces profils est que le niveau de bruit et de privilege creep croissent dans la même direction. Cette hypothèse est cohérente d'un point de vue temporel de dérive dans la gestion des accès.

Nom	**Acronyme**	**Bruit**			
		pourcentage	**densité**	**légitime**	**PC**
no noise no pc	NN	0	N/A	0%	0%
low noise, low density	LNLD	5%	1%	10%	3%
high noise, low density	HNLD	15%	1%	15%	5%
low noise, high density	LNHD	5%	4%	15%	5%
high noise, high density	HNHD	15%	5%	20%	8%
par défaut	/	15%	2%	10%	3%

Table 5.2 Profils de niveaux de bruit & privilege creep

Ces cinq profils servent à paramétrer l'ajout de bruit et de privilege creep aux permissions légitimes, décrit dans la section 4.3.2 et 4.3.3. Les valeurs utilisées sont renseignées dans le tableau 5.2. Enfin les paramètres manquants, c'est-à-dire la répartition des utilisateurs et des permissions sur les nœuds de la structure des permissions légitimes est donné par le cas par défaut renseigné sur la table 5.3 dans la partie suivante. Cette configuration par défaut correspond à une configuration sans tension un peu plus large sur les minimum et maximum de permissions par nœud. Les valeurs exactes pour ces profils sont choisies de sorte à rester cohérentes avec celles rencontrées dans la littérature.

5.1.2 Niveaux variables de tension sur les utilisateurs et permissions

De la même façon qu'on a défini des profils pour différents niveaux de bruit, on propose une deuxième expérimentation qui évalue la performance de l'approche proposée lorsque le nombre de permissions ou d'utilisateurs sont restreints. Cette idée est née d'une conjecture directe : la clusterisation pourrait être moins bonne lorsque le nombre d'utilisateurs est faible, et aussi lorsque les permissions communes aux utilisateurs se font rares. Afin d'évaluer la véracité de cette conjecture, on propose donc quatre profils de tension mise sur les utilisateurs et les permissions :

- aucune tension mise (no tension)
- tension sur les permissions
- tension sur les utilisateurs
- tension sur les utilisateurs et les permissions simultanément (tension sur les deux)

Nom	**Acronyme**	**Utilisateur**		**Permission**	
		min	**max**	**min**	**max**
no tension	NT	15	25	10	40
tension sur permissions	TP	15	25	2	6
tension sur utilisateurs	TU	2	8	10	40
tension sur les deux	TUTP	2	8	2	6
par défaut	/	15	25	15	45

Table 5.3 Profils de tension sur les utilisateurs et les permissions

Les valeurs exactes sont renseignées sur le tableau 5.3. Afin d'évaluer les capacités de détection de privilege creep et de correction de bruit, on donne pour ces différents profils la configuration par défaut renseignée sur la table 5.2 précédemment. Cette configuration par défaut est un compromis d'un bruit fort, mais avec une densité moyenne et un niveau de privilege creep assez faible.

5.2 Jeux de données réels

Les jeux de données du monde réel fournis par Ene et al. [1] sont utilisés pour l'évaluation. Il s'agit de : *americas_large, americas_small, apj, customer, domino, emea, firewall1, firewall2, healthcare*. Malgré leur publication en 2008, ces jeux de données sont devenus des références établies dans la littérature et continuent d'être utilisés dans des études récentes

pour la comparaison des performances sur des instances réelles [40, 42, 43, 55, 66]. On les pivote sous forme de matrice UPA avec une transformation expliquée dans la section 3.1.2.

5.3 Jeux de données réels du partenaire industriel

Le jeux de données fourni par le partenaire industriel se trouve sous forme tabulaire. Il est extrait d'une configuration Active Directory sous la forme d'une table relationnelle. Chaque ligne correspond à une association entre un compte utilisateur et une permission, tandis que chaque colonne représente une variable ou un attribut descriptif. La structure initiale de la table est composée des colonnes suivantes :

personal_id	account_id	group_id	group_domain	context_1	...	context_n

Où :

- **personal_id** correspond à un identifiant d'une personne physique réelle.
- **account_id** correspond à un identifiant de compte Active Directory. Une personne physique peut avoir plusieurs comptes liés à son identité.
- **group_id** correspond à un identifiant de groupe de sécurité Active Directory. Un groupe de sécurité s'applique au compte Active Directory.
- **group_domain** fait référence au domaine dans lequel le groupe de sécurité s'applique.
- $\textbf{context}_i$ renvoie à des informations supplémentaires sur la personne identifiée dans la ligne (titre du poste, responsable, service, etc.).

Ce tableau est d'abord réduit à deux champs principaux :

- **id** qui prend la valeur de **personal_id** pour correspondre à un utilisateur sur le système.
- **permission** représente un groupe de sécurité au sein d'un domaine spécifié. Essentiellement, cela correspond à **group_id@group_domain**. Une autre façon de définir ce champ est de restreindre l'effort d'extraction de rôle à un seul domaine, le champ est alors strictement égal à la valeur du champ **group_id**.

Il est possible de choisir l'identifiant du compte (**account_id**) plutôt que l'identifiant personnel (**personal_id**). Cependant, dans le cadre du role mining, il est plus pertinent de regrouper

toutes les permissions sous l'identité unique d'une personne plutôt que sous différents comptes gérés par une solution de GIA. En effet, il est possible pour un utilisateur de changer de compte tout en conservant l'ensemble de ses permissions. On forme alors une table relationnelle réduite avec la structure de colonnes suivante :

id	permission

Les valeurs contextuelles sont stockées dans une table avec cette structure de colonne :

id	context_1	...	context_n

La dernière étape du prétraitement consiste à faire pivoter la table relationnelle id-permission afin d'obtenir une matrice UPA, qui représente le format standard des données utilisées en role mining. Cette transformation est expliquée en détail dans la section 3.1.2.

5.4 Métriques

L'objectif principal des métriques choisies est de mesurer la précision de détection des instances de privilege creep et leur correction tout en préservant l'expression des permissions légitimes. Les métriques retenues s'inspirent des travaux de la littérature, présentées dans le tableau 2.2. On introduit les métriques suivantes :

1. Rétention des Permissions Légitimes (RPL) : La F-mesure (Voir l'équation 2.7) calculée entre la matrice UPA nettoyée (CUPA) et l'UPA légitime de référence, en excluant les outliers. Une valeur élevée reflète la précision de la rétention des permissions légitimes, tout en minimisant la rétention du bruit et du privilege creep mal identifié.

2. Expression des Permissions (EP) : Le nombre d'assignations de permissions conservées dans l'UPA nettoyée (CUPA) divisé par le nombre d'assignations dans l'UPA bruitée. Un processus de nettoyage efficace donne une valeur d'expression des permissions se rapprochant de celle calculée sur la matrice légitime.

3. Correction du Privilege Creep (CPC) : Le pourcentage d'assignations affectées par le privilege creep qui ont été efficacement retirées. C'est une nouvelle métrique introduite. Une valeur proche de 1 reflète l'efficacité du processus de nettoyage. Une valeur de 0 indique qu'aucune assignation issue de privilege creep a été retirée.

4. Précision de la Détection du Privilege Creep (PDPC) : Le F-mesure (Voir l'équation 2.7) calculé entre l'ensemble des outliers identifiées et l'ensemble des utilisateurs anormaux de référence. C'est une mesure de l'exactitude de la détection des utilisateurs atteints de privilege creep.

5. Écart du Nombre de Rôles (ENR) : La différence entre le nombre de rôles idéaux, identifié comme étant le nombre de feuilles dans le modèle (voir section 4.3.1) et le nombre de rôles extraits, divisée par le nombre de rôles idéaux.

6. Proportions de privilege creep détecté : Pour chaque type de privilege creep rajouté sur les jeux de données synthétiques, on relève la proportion d'instances correctement identifiées par rapport au nombre réel d'instances rajoutées. Cela permet d'avoir une vue plus large sur la détection de privilege creep et de voir quel type est plus facilement détectable.

7. Temps d'exécution : Le temps d'exécution du processus de nettoyage en secondes.

8. Nombre de rôles : exclusivement utilisé pour les jeux de données réels.

Le tableau 5.4 résume les inspirations et sources des métriques retenues, ainsi que leur application sur des jeux de données synthétiques ou réelles. En effet, RPL, CPC, PDPC, ENR et les proportions de privilege creep détectés, nécessitent l'existence d'une connaissance de référence (ground truth). Elles ne peuvent donc être calculées que sur les jeux de données synthétiques.

Métrique	**Inspiration/Source**	**utilisation données réelles**	**utilisation données synth.**
Rétention des Permissions Légitimes (RPL)	Nouvelle métrique qui prend inspiration de la robustesse au bruit de Vaidya et al. (voir le tableau 2.2). L'accent est mis sur les permissions légitimes, puisqu'on veut non seulement retirer le bruit, mais aussi le privilege creep tout en retenant le maximum de permissions légitimes. La F-mesure assure que le déséquilibre de classe entre permissions légitimes et illégitimes n'influe pas sur les mauvais résultats	✗	✓
Expression des Permissions (EP)	Métrique usuelle en role mining inexact, souvent formulée différemment avec l'erreur de reconstruction $E_{reconstruction}$. On préfère utiliser cette version, car sur les jeux de données réels, on peut mettre en lumière l'influence des outliers sur l'expression de permission d'une façon plus interprétable.	✓	✓
Correction du Privilege Creep (CPC)	Nouvelle métrique introduite qui permet, avec la rétention de permission légitime, de quantifier si les assignations issues de privilege creep sont effectivement bien retirées.	✗	✓
Précision de la Détection du Privilege Creep (PDPC)	Nouvelle métrique qui prend inspiration sur l'Exactitude de Parkinson et al. (voir le tableau 2.2). On utilise la F-mesure afin de ne pas biaiser les résultats à cause du déséquilibre de nombre entre les instances de privilege creep et les utilisateurs aux permissions légitimes (voir explication dans la section 2.5.4)	✗	✓
Écart du Nombre de Rôles (ENR)	Nouvelle métrique adaptée du nombre de rôles, donne plus d'information vis-à-vis de l'objectif de role mining sur les jeux de données synthétiques lorsqu'on s'attend à un nombre de rôles précis.	✗	✓
Proportions de privilege creep détecté	Nouvelle métrique introduite puisqu'on a différents scénarios pour les instances de privilege creep	✗	✓
Temps d'exécution et nombre de rôles	Métriques usuelles de la littérature	✓	✓

Table 5.4 Récapitulatif des métriques d'évaluation choisies

Pour les jeux de données réels, on relève également le nombre d'outliers détectés afin de le comparer au nombre total d'utilisateurs. L'expression des permissions est calculée de deux manières : avec et sans les outliers, ce qui permet de mieux interpréter le cas des outliers sur certains jeux.

CHAPITRE 6 RÉSULTATS

Ce chapitre détaille les résultats obtenus avec le cadre d'évaluation décrit précédemment. On commence par vérifier la validité des jeux de données synthétiques servant l'évaluation. Ensuite, on présente et interprète les résultats d'expérimentation sur les jeux de données synthétiques générés et les jeux de données réels de la littérature. On propose également une analyse comparative de la performance opérationnelle de la méthode développée, évaluée sur des jeux de données réels fournis par le partenaire industriel. Enfin, le chapitre se conclut par une discussion générale sur les résultats observés.

6.1 Validation

Avant de présenter les résultats, on veut s'assurer que ceux-ci sont valides. En particulier, on veut s'assurer que la structure des jeux de données synthétiques produits correspond à une structure de jeu de données réel.

6.1.1 Méthode de validation

Pour procéder à la validation, on part d'un constat sur les jeux de données réels : la distribution du partage des permissions, c'est-à-dire le pourcentage d'utilisateurs portant chaque permission, suit une courbe avec une forme particulière. Il en est de même pour la distribution de la quantité de permissions détenues par les utilisateurs.

En règle générale, on observe que 70% à 80% des permissions ne sont partagées que par un petit groupe d'utilisateurs tandis que les 30% à 20% restants concentrent la quasi-totalité des assignations et sont grandement partagées. On aimerait retrouver cette tendance sur les jeux de données générés synthétiquement.

Pour s'en rendre compte, on calcule deux quantités :

- Le nombre d'utilisateurs qui possède chaque permission.
- Le nombre de permissions que chaque utilisateur détient.

On vient ensuite ordonner par ordre croissant ces valeurs et normaliser par le nombre total d'utilisateurs ou de permissions. Cette transformation revient respectivement à sommer la matrice UPA sur les colonnes ou les lignes et trier par ordre croissant les valeurs obtenues avant de les normaliser avec les dimensions de la matrice.

On nomme respectivement les courbes obtenues :

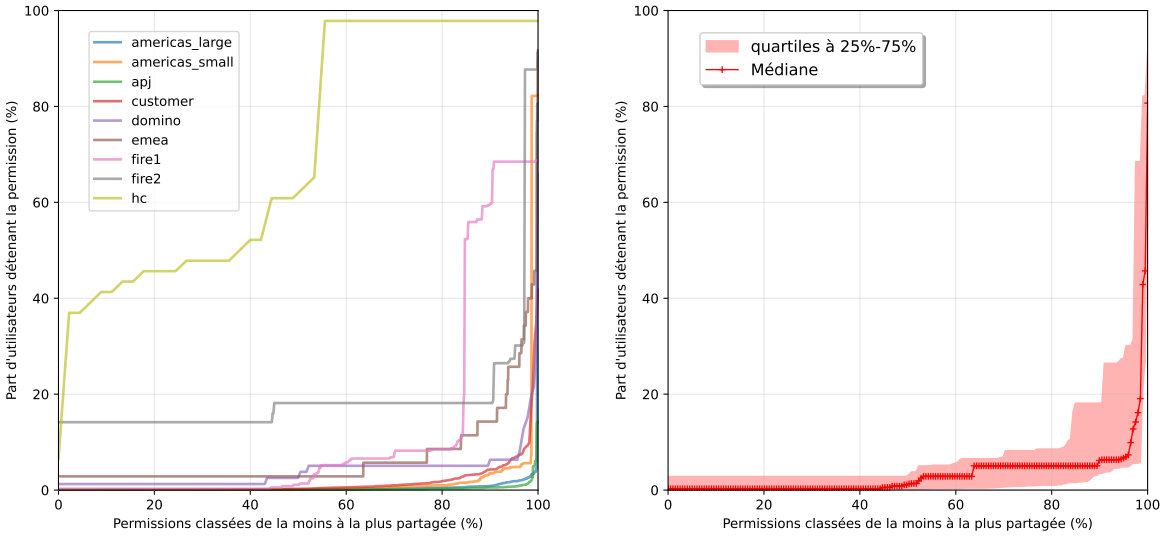
- La courbe de concentration des utilisateurs sur les permissions (UsP).
- La courbe de concentration des permissions sur les utilisateurs (PsU).

Ce processus est appliqué aux jeux de données issus de la littérature ainsi qu'à l'échantillon de jeux de données synthétiques généré pour l'évaluation. La comparaison des courbes obtenues constitue une validation empirique : leur ressemblance atteste du réalisme et de la fidélité des données synthétiques par rapport aux données réelles.

6.1.2 Comparaison

Pour faciliter la comparaison générale, on va utiliser les courbes de concentration médianes qui montrent les tendances sur les jeux de données réels. Ainsi sur les figures 6.1 et 6.2, on montre les courbes UsP et PsU superposées des jeux de données réels ainsi que la courbe médiane avec les quartiles à 25% et 75%.

Il est important de noter que bien que ces courbes soient similaires, elles représentent deux visions bien différentes des jeux de données. Avec la courbe UsP médiane en figure 6.1b On retrouve par ailleurs le constat classique de la GIA qui est que 70% à 80% des permissions sont partagées par un petit groupe d'utilisateurs tandis que les 30% à 20% restants concentrent la quasi-totalité des assignations à exprimer en RBAC. Les permissions les plus partagées sont par ailleurs partagées par au moins 80% des utilisateurs dans plus de la moitié des cas.

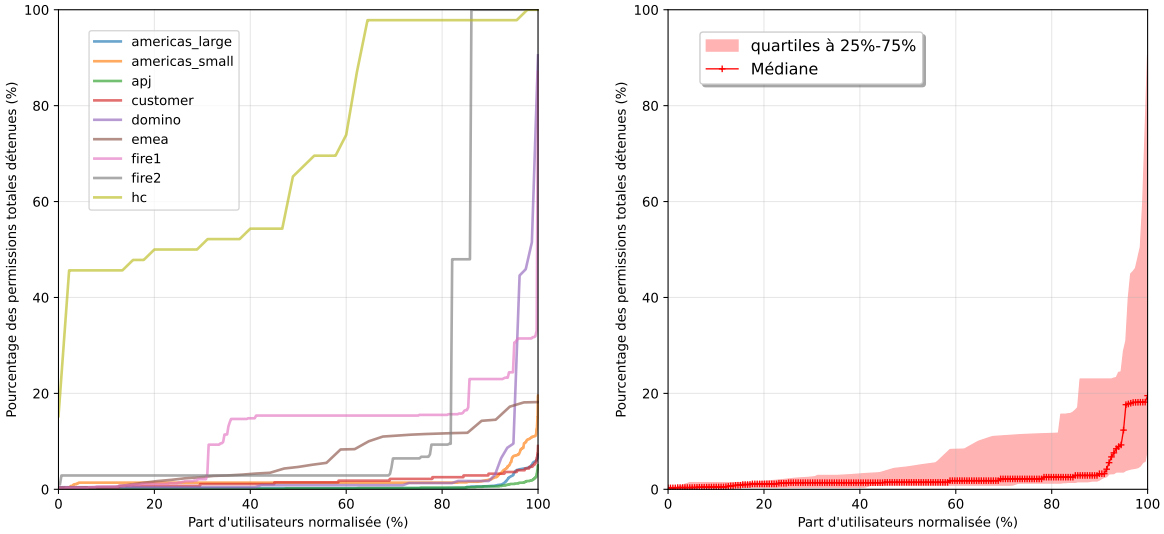


(a) Courbes multiples

(b) Courbe médiane

Figure 6.1 Courbes de concentration UsP des jeux de données réels

Sur la courbe PsU en figure 6.2b une tendance similaire peut être observée : les 20% d'utilisateurs ayant le plus de permissions concentrent la majeure partie des permissions octroyées. Cependant, on observe qu'en moyenne les utilisateurs ayant le plus de permissions possèdent rarement plus de 20% du nombre de permissions totales. Cette propriété se vérifie pour les jeux de données les plus volumineux comme *americas_large, americas_small, customer, apj*. Les jeux de données qui échappent à cette règle possèdent au moins un utilisateur administrateur qui concentre la quasi-totalité des permissions comme *domino, firewall1, firewall2, healthcare*. On notera aussi à la vue des figures 6.1 et 6.2 que le jeu de données *healthcare* est un net outlier vis-à-vis des courbes de concentration et de la petitesse du jeux de données (46 utilisateurs pour 46 permissions différentes, cf. 2.3.1).



(a) Courbes multiples

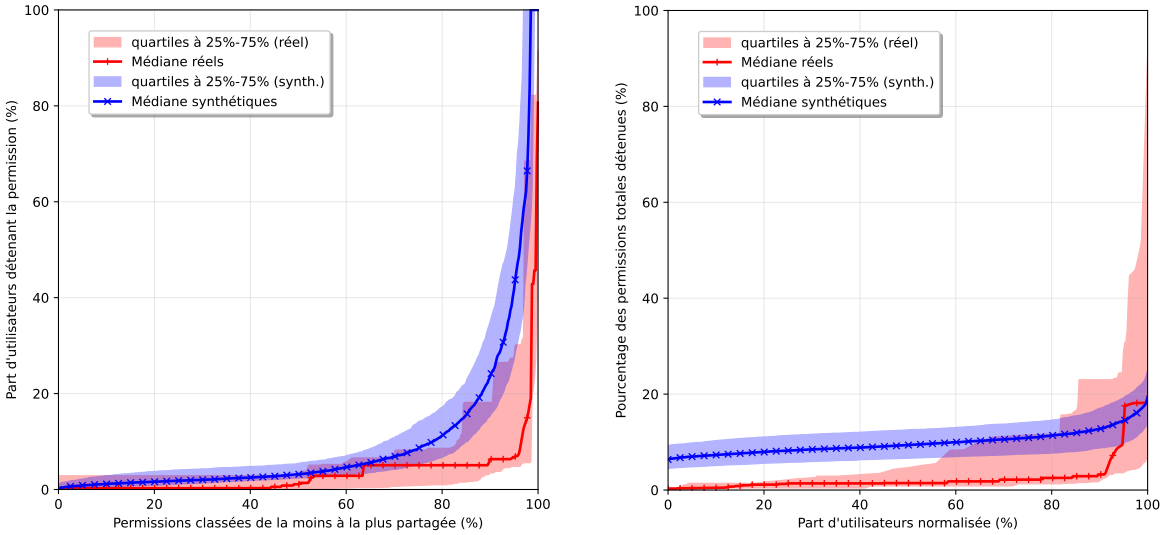
(b) Courbe médiane

Figure 6.2 Courbes de concentration PsU des jeux de données réels

On trace désormais les courbes de concentration UsP et PsU médianes sur les jeux de donnés synthétiques générés pour l'évaluation, soit un échantillon de 1080 jeux de données, le détail de ce nombre est donné dans la section suivante 6.1.3. Sur la figure 6.3 on superpose les courbes obtenues sur les jeux de données réels et sur les jeux de données synthétiques.

La comparaison des courbes UsP révèle une forte similitude entre les courbes des données réelles et des données synthétiques. Les deux courbes présentent une allure similaire et leurs quartiles respectifs se superposent, attestant d'une reproduction possible de la concentration des utilisateurs sur les permissions. La courbe synthétique est cependant plus lisse du fait de la taille de l'échantillon considéré.

Les courbes PsU révèlent en revanche une divergence plus marquée. On observe un écart quasi constant d'environ 10% entre les deux distributions, accompagné d'une séparation des quartiles sur approximativement la moitié du domaine. Cependant, on est rassurés de voir que le maximum atteint par la courbe PsU en moyenne se situe autour de 20%, comme sur la courbe des jeux de données réels.



(a) UsP réel vs synthétique

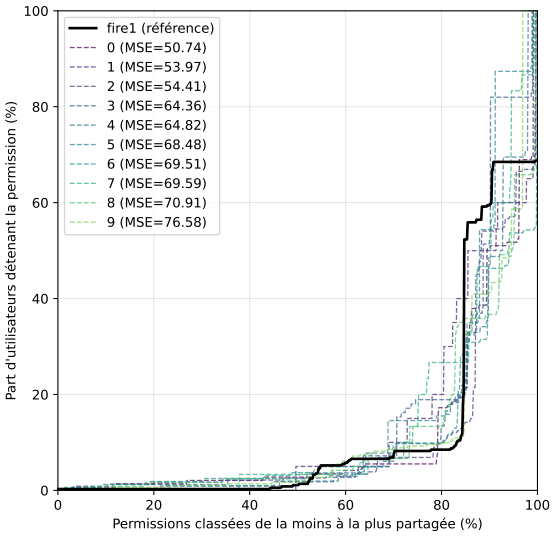
(b) PsU réel vs synthétique

Figure 6.3 Comparaison des courbes de concentration médianes des jeux de données

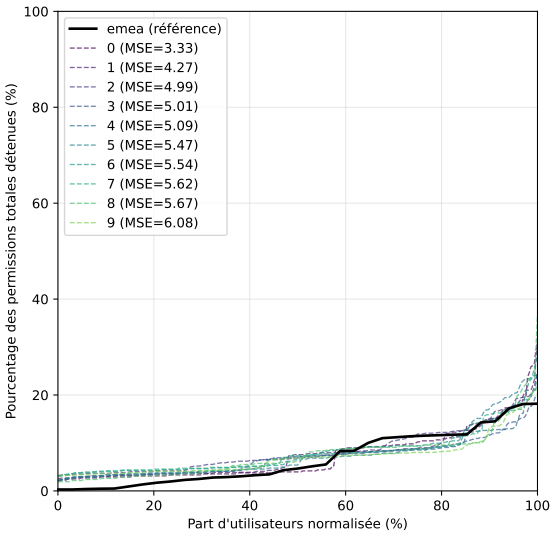
Ces observations permettent de conclure que les jeux de données synthétiques reproduisent en moyenne fidèlement les distributions de concentration observées dans les jeux de données réels de la littérature. Si les courbes médianes PsU présentent des divergences plus marquées, celles-ci peuvent s'expliquer par la taille limitée de l'échantillon de données réelles, source potentielle de biais dans l'estimation des distributions.

Cette conclusion est d'autant plus motivée qu'on peut trouver des jeux de données synthétiques dont les distributions UsP et PsU sont très proches d'un jeu réel donné, au sens des moindres carrés (MSE). Sur la figure 6.4, on renseigne quatre exemples des 10 courbes UsP ou PsU les plus proches des distributions de jeux de données réels au sens des moindres carrés.

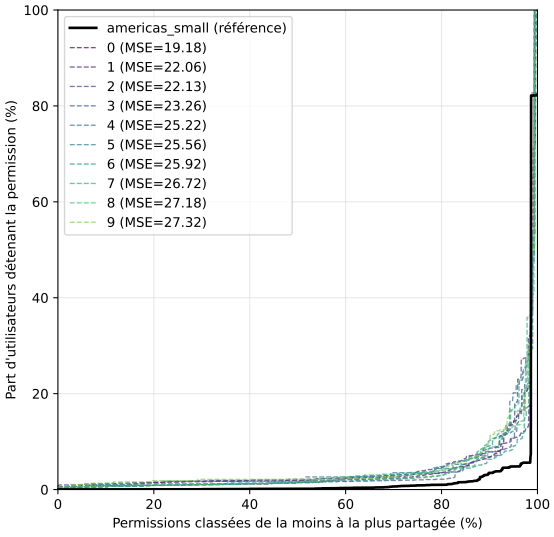
Ceci achève la validation, car on a établi une concordance statistique globale entre données réelles et synthétiques, mais on démontre également la capacité du processus de génération des données à reproduire fidèlement les caractéristiques spécifiques de jeux de données individuels issus de contextes organisationnels variés.



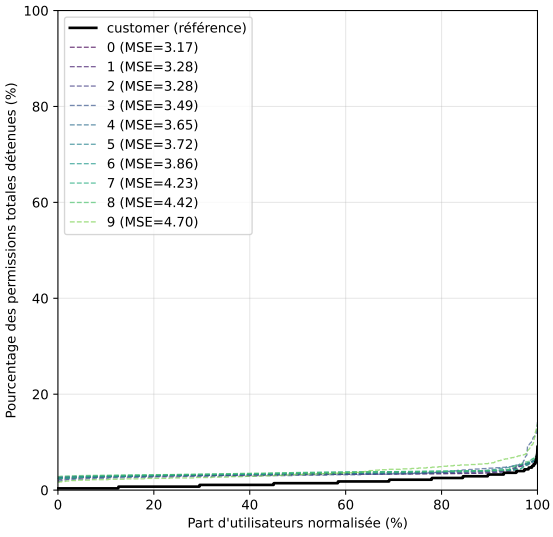
(a) UsP *firewall1*



(b) PsU *emea*



(c) UsP *americas_small*



(d) PsU *customer*

Figure 6.4 Exemples de courbes de concentration de jeux de données réels vs synthétiques

6.1.3 Cadre d'évaluation sur données synthétiques

Les résultats d'expérimentation sont présentés sous forme de boîtes à moustaches, indiquant la médiane, les quartiles à 25% et 75% et les valeurs considérées comme aberrantes pour les métriques représentées. Chaque graphique illustre la distribution d'une métrique donnée, pour différents profils (de bruit ou de tension) utilisés pour la génération des données.

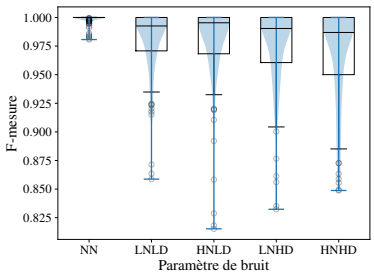
Chaque couple de profils bruit-structure et tension-structure est utilisé 20 fois pour générer 20 jeux différents. Étant donné qu'il y a six paramètres de structure, chaque boite à moustache décrit donc 120 échantillons différents. Ainsi par exemple, la boite à moustache "LNLD" de la figure 6.5a, décrit les résultats de rétention de permissions légitimes de la méthode proposée sur 120 jeux de données synthétiques différents générés avec le profil de bruit LNLD.

Pour l'évaluation en fonction du bruit et du privilege creep, cela veut dire que 600 échantillons sont représentés sur chaque graphique. Pour l'évaluation en fonction de la tension, ce sont 480 échantillons qui sont représentées par graphique.

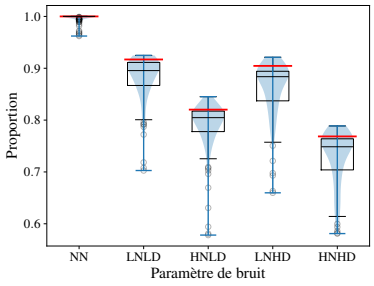
6.2 Sur jeux de données synthétiques

6.2.1 Niveaux variables de bruit et de privilege creep

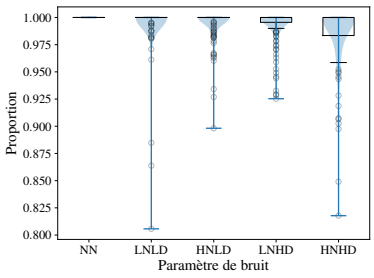
La médiane pour l'EP idéale est indiquée par une ligne rouge sur la figure 6.5b.



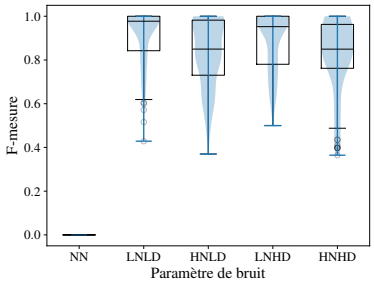
(a) Rétention des Permissions Légitimes (RPL)



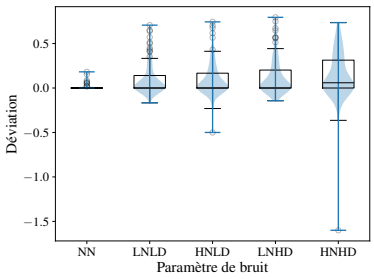
(b) Expression des Permissions (EP)



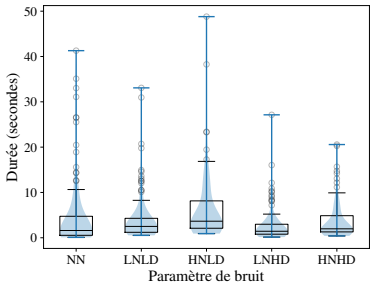
(c) Correction du Privilege Creep (CPC)



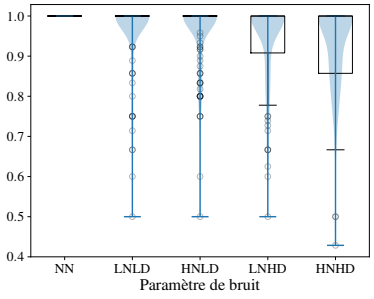
(d) Précision de la Détection du Privilege Creep (PDPC)



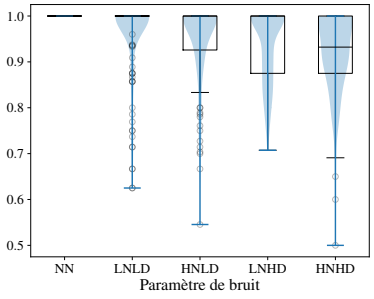
(e) Écart du Nombre de Rôles (ENR)



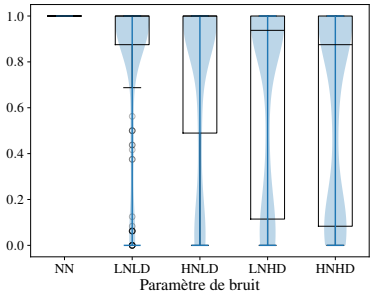
(f) Temps d'exécution



(g) Proportion de PC total identifié



(h) Proportion de PC partiel identifié



(i) Proportion de PC temporaire identifié

Figure 6.5 Performances sur différents niveaux de bruit et privilege creep

Interprétation

Les figures 6.5a et 6.5b montrent simultanément que les assignations de permissions légitimes ont été récupérées avec précision dans la majorité des cas, évitant l'expression d'assignations bruitées ou issues de privilege creep. En effet, la RPL médiane est supérieure à 97,5% ce qui veut dire qu'en moyenne, on a classifié efficacement les permissions retenues comme légitimes.

Pour ce qui est de l'EP, on observe les variations attendues en comparant la médiane d'EP obtenue par rapport au cas idéal : l'EP diminue à mesure que le bruit et les instances de privilege creep sont plus importantes. Cependant, on voit que notre algorithme a tendance à exprimer moins de permissions en moyenne que le cas idéal, en particulier sur les jeux bruités. Cette tendance semble ne pas être affectée par les niveaux de bruit, en effet lorsqu'on calcule la différence d'EP entre la sortie de l'algorithme et le cas idéal, on obtient la table 6.6.

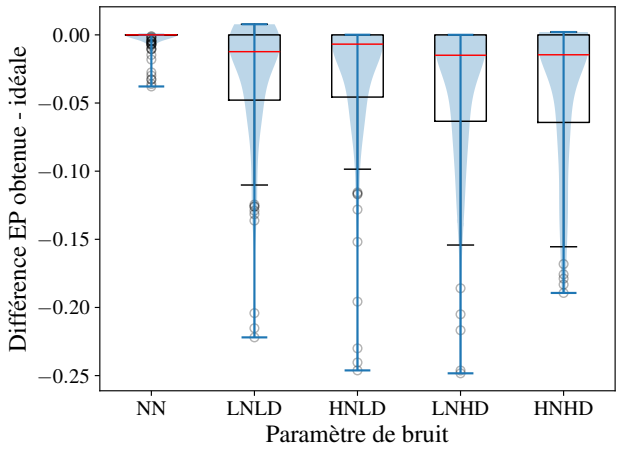


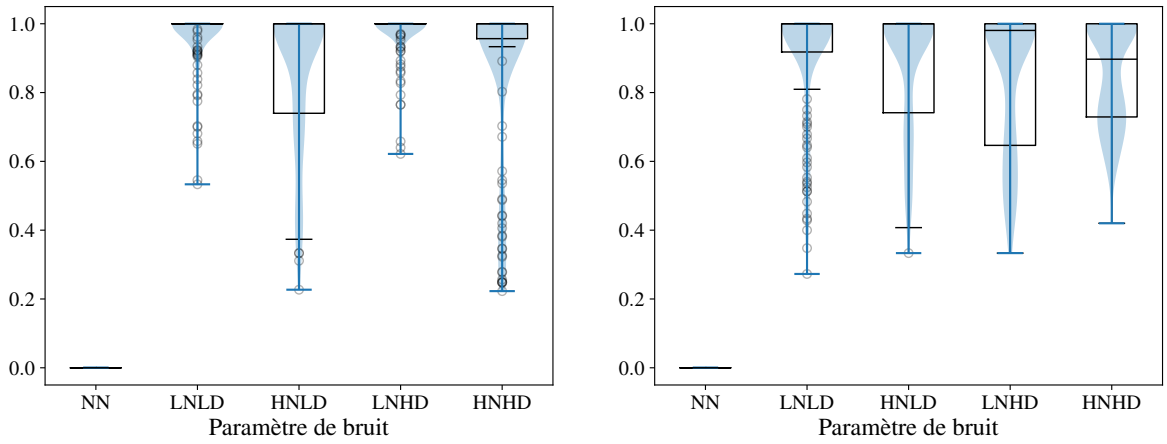
Figure 6.6 Différence d'EP entre la sortie de l'algorithme de RM et le cas idéal en fonction du bruit

On observe donc qu'en moyenne, l'algorithme exprime 2% de permissions en moins que le cas idéal. Les cas les plus extrêmes dévient de 20%, ce qui est cohérent avec les cas extrêmes de basse RPL. On comprend donc que dans ces cas isolés, l'algorithme a révoqué des permissions légitimes lors du nettoyage.

Concernant le privilege creep, des informations contradictoires semblent provenir des figures 6.5c et 6.5d : la F-mesure de détection oscille autour de 80%, atteignant parfois des valeurs inférieures à 50% lors des pires exécutions, tandis qu'une majorité d'assignations issues du privilege creep sont corrigées avec des valeurs de CPC médianes à 100% sur tous les niveaux de bruit. Pour comprendre ce phénomène, il faut raisonner de manière statistique et observer les graphes 6.5g 6.5g et 6.5i. En effet, on observe que les instances de privilege creep de

type I sont en moyenne mieux détectées que les instances de type II, qui rejoignent alors des clusters valides. Or, comme les assignations de permission de privilege creep de type II sont des permissions additionnelles uniquement valables pour les membres d'un projet, elles sont aisément retirées avec le processus de nettoyage lorsque les autres utilisateurs d'un cluster ne font pas partie du projet. Enfin, on remarque aussi que la détection de privilege creep est plus ardue lorsque la quantité de bruit augmente, puisque la PDPC est bien moins bonne dans les cas HNLD et HNHD (beaucoup de bruit indépendamment de la densité) sur la figure 6.5d.

Une dernière observation à faire sur la PDPC est de regarder la précision et le rappel qui la composent afin de comprendre clairement comment on a détecté les instances de privilege creep. En regardant les figures de précision 6.7a et de rappel 6.7b On se rend compte d'un bon équilibre, bien que la précision soit en moyenne légèrement plus élevée. Cela indique qu'on a en moyenne un peu plus de faux négatifs que de faux positifs avec les paramètres déterminés automatiquement. Il est probable qu'avec du fine-tuning, on parvienne à faire augmenter le rappel sans détériorer la précision en moyenne.



(a) Précision PDPC

(b) Rappel PDPC

Figure 6.7 Comparaison entre la Précision et le Rappel pour la PDPC en fonction du bruit

La figure 6.5e révèle que l'approche proposée tend à produire moins de rôles qu'attendu, probablement en raison de permissions légitimes étant effacées par le processus de nettoyage.

Le temps d'exécution moyen pour le processus de nettoyage est inférieur à 5 secondes, ne dépassant jamais une minute comme le démontre la figure 6.5f. Des temps d'exécution plus longs peuvent être attribués à des jeux de données exceptionnellement volumineux. En effet, lorsqu'on représente graphiquement les données de temps d'exécution (figure 6.8) en fonction des profils de génération de permissions légitimes, on se rend compte que les valeurs les plus élevées sont détenues par les profils produisant les plus gros jeux de données.

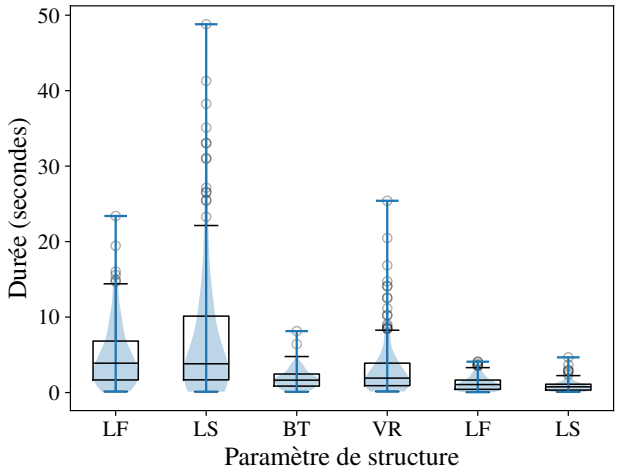
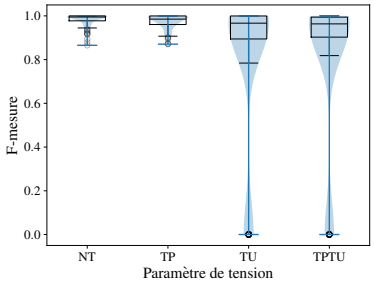


Figure 6.8 Temps d'exécution en fonction du profil de génération

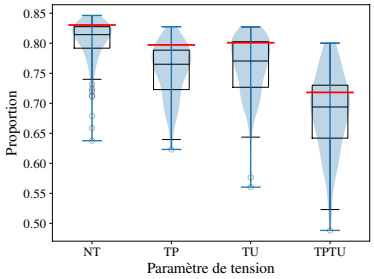
Lorsque les jeux de données sont exempts de bruit et de privilege creep, les figures 6.5b, 6.5a et 6.5e confirment que l'approche proposée récupère avec précision les assignations légitimes sans dégrader le jeux de données, en exprimant la quasi-totalité des permissions présentes, et trouve un nombre de rôles en moyenne très proche du cas idéal (pas de déviation du tout pour 95% des cas). On a aussi très peu d'outliers identifiés voir aucun dans la majorité des cas. Toutes ces informations portent à croire que l'approche proposée ne dégrade pas les jeux de données de contrôle d'accès lorsque celui-ci ne présente pas d'anomalies.

6.2.2 Niveaux variables de tension sur les utilisateurs et permissions

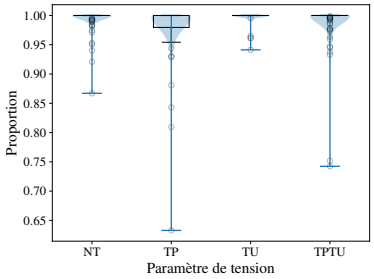
La médiane pour l'EP idéale est indiquée par une ligne rouge sur la figure 6.9b.



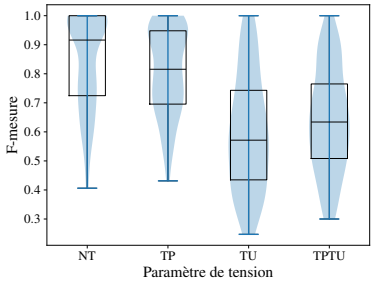
(a) Rétention des Permissions Légitimes (RPL)



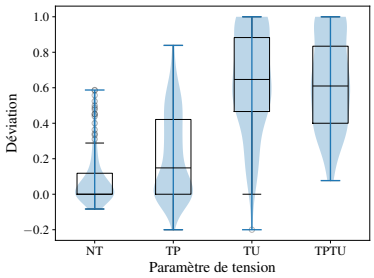
(b) Expression des Permissions (EP)



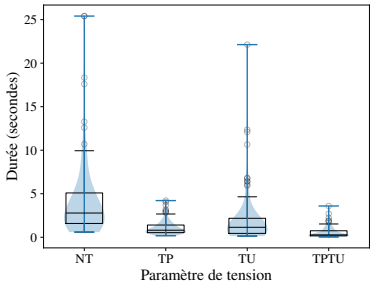
(c) Correction du Privilege Creep (CPC)



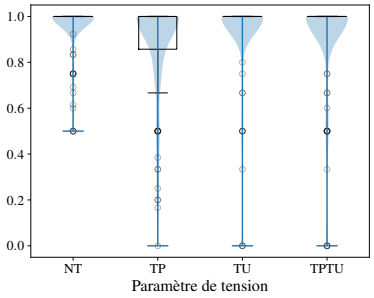
(d) Précision de la Détection du Privilege Creep (PDPC)



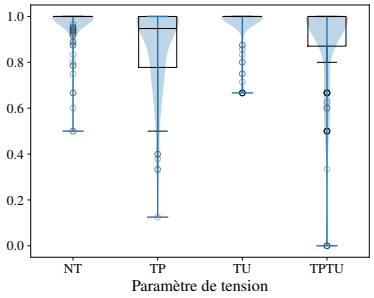
(e) Écart du Nombre de Rôles (ENR)



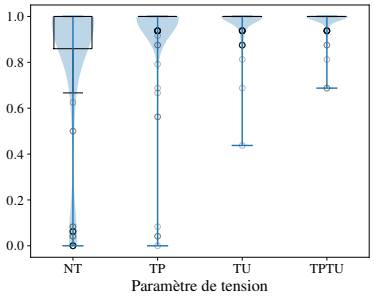
(f) Temps d'exécution



(g) Proportion de PC total identifié



(h) Proportion de PC partiel identifié



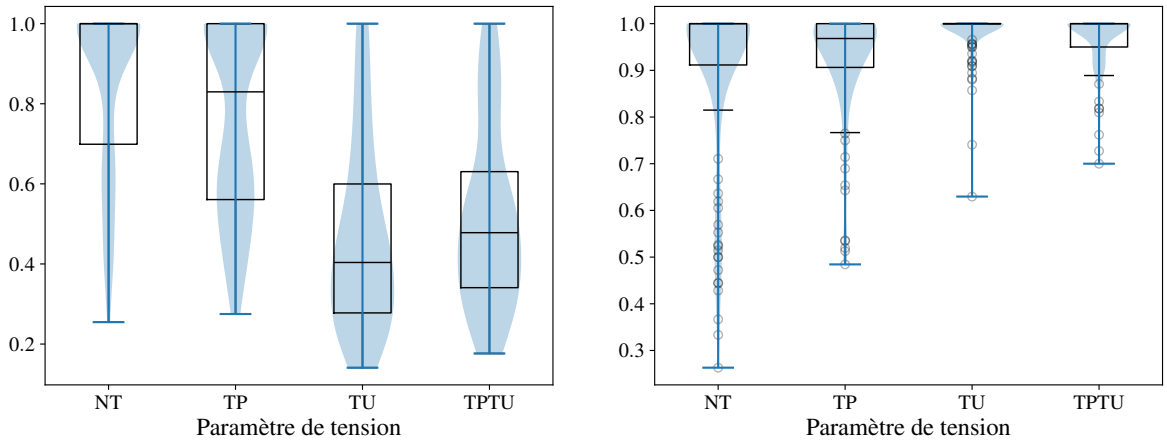
(i) Proportion de PC temporaire identifié

Figure 6.9 Performances sur différents niveaux de tension

Interprétation

Les figures 6.9a et 6.9b démontrent que les affectations de permissions légitimes ne sont généralement pas récupérées avec précision lorsqu'une tension est exercée sur les utilisateurs. En effet, la RPL chute à 0% sur plusieurs exécutions et est en moyenne plus faible dans les cas ou la tension est mise sur les utilisateurs. L'EP est plus faible qu'attendue aussi, on voit des écarts plus grands avec la médiane d'EP que sur l'expérience précédente. Le constat est le même et plus fort que la dernière fois : les affectations légitimes ont été massivement supprimées.

Ce phénomène peut s'expliquer en partie avec l'étape de clustering, notamment en regardant la PDPC sur la figure 6.9d. La F-mesure diminue de 30% environ lorsque la tension est mise sur les utilisateurs. Pour mieux comprendre ce qu'il se passe, il faut regarder les métriques de précision et de rappel utilisées pour calculer la PDPC sur les figures 6.10a et 6.10b. Ici par rapport à l'expérience précédente, c'est la précision qui est mauvaise et le rappel qui semble plutôt bon. Pour une majorité des exécutions sous tension utilisateur, on a donc considéré trop d'outliers que ce qu'on devait : Il y a plus de faux positifs. (voir les figures 6.7a et 6.7b pour comparaison)



(a) Précision PDPC

(b) Rappel PDPC

Figure 6.10 Comparaison entre la Précision et le Rappel pour la PDPC en fonction de la tension

Cette observation est logique : le clustering devient inefficace dans le cas ou trop peu d'utilisateurs partagent les mêmes permissions, qui est une limite inhérente à la méthode.

En conséquence, le processus de nettoyage tend à supprimer un nombre plus élevé de permissions en raison de clusters mal identifiés, ce qui mène à une surcorrection du privilege creep

comme on peut le voir sur la figure 6.9c, ou la correction semble bonne. Idem pour les types de privilege creep identifiés sur les figures 6.9g, 6.9h et 6.9i, où on croit à une amélioration de la détection à mesure que le nombre d'utilisateurs diminue, interprétation illusoire au vu des autres informations données.

Ces effets se propagent à l'étape d'extraction des rôles, aucun rôle ne peut être extrait lorsque de grandes quantités de permissions ont été supprimées. La figure 6.9e illustre les cas où les exécutions sous tension utilisateur produisent en moyenne la moitié des rôles qui seraient normalement nécessaires pour exprimer les affectations (déviation avoisinant 60%), tandis que le cas témoin possède des performances similaires à la première expérience.

Sur la figure 6.9f, on observe que les temps d'exécution du nettoyage sont plus rapides que dans les expériences précédentes, avec une moyenne de 3 secondes en raison de la taille réduite des jeux de données sous tension utilisateur et/ou permission.

Ainsi donc, la tension sur les permissions semble avoir un impact minime sur la RPL et de PDPC, bien qu'un léger impact sur le role mining soit observé sur la figure 6.9e, où en moyenne 20% de rôles en moins sont produits. Cependant, la tension sur les utilisateurs représente un réel problème sur les performances de l'algorithme proposé.

On a observé sur les deux expériences précédentes que l'EP semblait ne pas varier ni en fonction des niveaux de bruit ni en fonction de la tension. En regardant l'EP en fonction des profils utilisés pour la génération de données sur la figure 6.11, on se rend compte que le problème provient principalement de la taille des jeux de données : plus le jeu de données est grand, plus il est difficile de l'exprimer. En effet, les profils générant les jeux de données les plus volumineux sont les profils large_flat, large_string et highly_random.

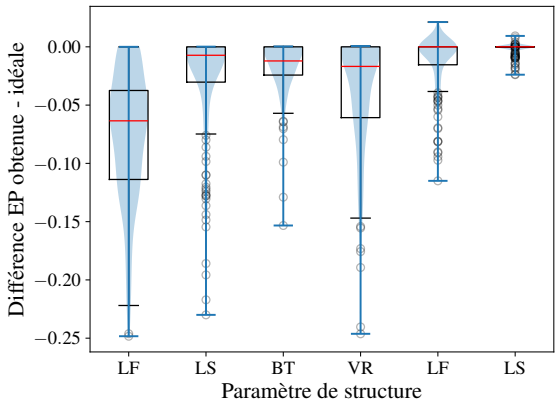


Figure 6.11 Différence d'EP entre la sortie de l'algorithme de RM et le cas idéal en fonction de la structure

6.3 Sur jeux de données réels

Les jeux de données réels permettent d'évaluer l'approche proposée dans un contexte où la connaissance de référence (ground truth) n'est pas disponible. Les jeux de données réels de la littérature évaluent principalement la capacité de l'approche à (1) maintenir une expression de permissions élevée après nettoyage, (2) produire un nombre de rôles réduit et gérable, et (3) identifier un nombre restreint d'utilisateurs outliers, permettant leur inspection manuelle pour validation. Ce sont ici les trois objectifs d'évaluation. Cette évaluation valide donc l'applicabilité pratique de l'approche dans des environnements réels.

Dans la mesure où l'ensemble des jeux de données réels est plus petit, les hyperparamètres sont choisis plus finement grâce à un processus expliqué dans les sections 3.1.1 et 3.1.1. Essentiellement, on vient fine-tuner r, le nombre de composantes pour TSVD, ϵ, le paramètre pour DBSCAN, et t_c, le seuil nettoyage pour les clusters.

Les valeurs pour ces hyperparamètres fine tunés sont rapportées dans le tableau 6.1 en même temps que la variance expliquée cumulée après réduction de dimension.

Jeu de données	r	$TVar$	ϵ	t_c
americas_large	10	0.646	2	0.3
americas_small	7	0.807	1	0.5
apj	6	0.350	0.5	0.005
customer	12	0.500	1	0.02
domino	4	0.840	1	0.1
emea	12	0.815	16	0.05
firewall1	3	0.882	0.3	0.5
firewall2	1	0.832	0.03	0.5
healthcare	3	0.841	1.5	0.4

Table 6.1 Hyperparamètres déterminés pour les jeux de données réels

Pour une majorité des jeux de données (*americas_small, domino, emea, firewall1, firewall2* et *healthcare*), la condition sur la variance totale cumulée donnée par l'équation 3.6 est vérifiée avec un nombre r de composantes inférieur à 12, en moyenne 5. Cependant, lors de la recherche de r pour les jeux *americas_large, customer* et *apj* ; la condition sur la composante courante explorée, donnée par l'inéquation 3.7 arrête le processus. La cause principale est la taille de ces jeux de données, qui font partie des 4 plus gros de ce benchmark (Cf. tableau 2.1), mais aussi à la distribution des assignations de permissions sur les utilisateurs.

Ensuite, le paramètre ϵ est fine tuné en prenant en compte la suggestion de l'algorithme NDCT. Ce paramètre est ajusté notamment lorsque le nombre d'utilisateurs est relativement faible comme sur les jeux de données *healthcare, emea* ou *domino*, car la technique automatique a tendance à donner une valeur de ϵ qui marque un trop grand nombre d'utilisateurs comme des outliers après clustering (de l'ordre de la moitié du nombre d'utilisateurs total). On réajuste donc ce paramètre pour viser 5 à 10% d'outliers identifiés par rapport au nombre total d'utilisateurs quand la méthode automatique échoue.

Enfin le paramètre t_c est quant à lui fixé à 0,5 par défaut. Lorsque ce seuil donne des niveaux d'EP trop bas (de l'ordre de moins de 10% de permissions exprimées) on considère qu'il faut le diminuer. C'est pour cela que les jeux de données *americas_large, healthcare* et *domino* se voient attribuer des seuils plus bas. Les seuils des jeux de données *apj, customer* et *emea* sont cependant anormalement bas. Deux effets peuvent expliquer ce résultat :

1. Un petit nombre d'utilisateurs concentre la quasi-totalité des assignations, ce sont des administrateurs et ils ont été retirés à l'étape du clustering, ce qui fait que l'ensemble des permissions restantes est minime. C'est le cas pour les jeux *emea* et *domino*.

2. Le clustering a regroupé un gros nombre d'utilisateurs sans réel motif de permissions communes entre eux, ce qui fait que le seuil requis pour ne pas retirer les permissions légitimes est anormalement bas. C'est le cas pour les jeux *apj, customer* et dans une moindre mesure *americas_large* et *healthcare*.

Les seuils t_c renseignés donnent des valeurs de Permissions Exprimées (PE) supérieures avoisinant 80% ou plus (voir tableau 6.2), afin d'assurer un role mining efficace.

Le Tableau 6.2 compile les résultats de l'approche. Le nombre optimal de rôles n_{optimal} utilisé pour la comparaison est fourni par HP Labs [1] et rapporté par Blundo et al. [42]. $n_{\text{rôles}}$ représente le nombre de rôles minés avec la méthode proposée dans cette recherche. Π représente l'expression de permission régulière définie dans la section 5.4, et π l'expression de permission calculée sans les outliers. Le nombre d'outliers détectés (nb out.) est rapporté aux côtés du nombre d'utilisateurs |U| de chaque jeu de données pour calculer le pourcentage d'outliers détectés parmi les utilisateurs (% out.).

Jeu de données	Π	π	|U|	nb. out	% out.	$n_{\text{rôles}}$	n_{optimal}
americas_large	0.755	0.841	3,485	45	1.29%	98	398
americas_small	0.881	0.913	3,477	31	0.89%	21	178
apj	0.727	0.759	2,044	13	0.64%	183	453
customer	0.798	0.812	10,021	58	0.58%	47	276
domino	0.171	0.919	79	7	8.86%	6	20
emea	0.615	0.799	35	4	11.43%	25	34
firewall1	0.905	0.974	365	14	3.84%	4	64
firewall2	0.994	1.000	325	7	2.15%	1	10
healthcare	0.881	0.946	46	4	8.70%	2	14

Table 6.2 Résultats sur des jeux de données du monde réel

Les jeux de données *emea* et surtout *domino* obtiennent des valeurs de permissions exprimées très différentes en fonction qu'on prenne en compte ou non les outliers. En effet, on a précédemment identifié que ces jeux de données sont caractérisés par un petit nombre d'utilisateurs qui concentrent un grand nombre de permissions, et qui sont ensuite marqués comme des outliers. L'approche proposée produit plus de 72% d'EP sur tous les autres jeux de données.

En examinant la différence entre Π et π, il est clair que sur des jeux de données comme *domino* et *emea*, le petit groupe d'outliers identifiées concentre la majorité des assignations de permissions. En effet, π est supérieur à 80%, ce qui signifie que les utilisateurs appartenant à un cluster ont 80% de leurs permissions exprimées en moyenne lorsqu'on ne considère pas les outliers dans le compte. Ces utilisateurs marqués comme des outliers sont probablement des administrateurs avec un nombre exceptionnellement élevé d'assignations de permissions, ce qui expliquerait en partie pourquoi ils ont été détectés comme tel.

Dans l'ensemble, le nombre d'outliers identifiées se situe entre 0.5% et 12% du nombre d'utilisateurs pour tous les jeux de données, avec un nombre absolu gérable par un humain, qui ne dépasse jamais 60 utilisateurs.

De plus, l'approche a produit moins de rôles sur tous les jeux de données par rapport au nombre optimal utilisé pour l'expression complète du jeu de données, divisant efficacement le nombre de rôles par 4 en moyenne. Le nombre exceptionnellement faible de rôles produits sur *firewall1* et *firewall2* est probablement dû aux jeux de données eux-mêmes, formés d'un grand nombre d'utilisateurs similaires avec seulement quelques utilisateurs représentant des exceptions.

Évaluation critique des objectifs

Les résultats sur les jeux de données réels confirment partiellement l'atteinte des objectifs d'évaluation fixés.

Expression des permissions élevée (1) : cet objectif est globalement atteint, puisque sept jeux de données sur neuf maintiennent une expression de permissions $\Pi > 0.72$. Cependant, *domino* ($\Pi = 0.171$) et *emea* ($\Pi = 0.615$) présentent des résultats problématiques. En excluant les outliers, ces valeurs remontent à $\pi = 0.919$ et $\pi = 0.799$ respectivement, ce qui suggère que l'approche fonctionne correctement pour les utilisateurs réguliers, mais que la présence d'administrateurs fausse fortement la métrique globale. Cette limitation révèle que l'objectif est atteint conditionnellement : l'expression est élevée pour les utilisateurs non-outliers, mais la métrique agrégée s'avère trompeuse sur des jeux de données déséquilibrés.

Nombre de rôles réduit et gérable (2) : cet objectif est clairement atteint, avec une réduction moyenne du nombre de rôles par un facteur 4 par rapport au nombre optimal. Les résultats pour *firewall1* (4 rôles vs 64) et *firewall2* (1 rôle vs 10) sont particulièrement marquants, bien que probablement attribuables à la structure particulière de ces jeux de données plutôt qu'à une performance supérieure de l'approche.

Identification d'un nombre restreint d'outliers (3) : cet objectif est partiellement atteint. Le nombre absolu d'outliers reste gérable (≤ 60 utilisateurs), mais le pourcentage varie considérablement (0.58% à 11.43%). Pour les petits jeux de données (*emea, domino, healthcare*), le taux d'outliers dépasse largement l'objectif de 5 à 10% initialement fixé lors du fine-tuning de ϵ. Cette inconsistance suggère que l'approche manque de robustesse face aux jeux de données de petite taille ou fortement déséquilibrés.

En synthèse, l'approche démontre son applicabilité pratique sur des jeux de données réels de taille moyenne à grande, mais révèle des limitations sur les petits ensembles de données ou ceux présentant de fortes concentrations de permissions. L'absence de connaissance de référence empêche toutefois de valider définitivement la pertinence des outliers détectés.

6.4 Sur jeux de données réels du partenaire industriel

Chez le partenaire industriel, la définition des rôles métiers constitue une étape essentielle dans la gestions des identités et des accès. Les rôles décrivent les responsabilités professionnelles des utilisateurs, et doivent regrouper l'ensemble des permissions nécessaires pour refléter fidèlement leurs activités.

Dans la pratique, chez le partenaire, la définition de tels rôles repose sur des approches combinant des analyses statistiques et une validation experte, visant à identifier des ensembles de permissions représentatifs pour des groupes homogènes d'utilisateurs. Cette approche, bien que pratique et efficace, présente de défis en termes de scalabilité, ce qui justifie l'exploration de méthodes plus intelligentes et automatiques, telles que celles proposées dans ce travail.

La méthode de role mining proposée est évaluée en comparaison de cette méthode opérationnelle. Des recommandations de permissions sont formulées avec l'information statistique sur les clusters. On attribue aux permissions un score égal à la prévalence de cette permission au sein d'un cluster. On obtient alors un classement des permissions par ordre de prévalence qu'on va comparer aux propositions humaines faites avec l'outil d'origine. On ne peut utiliser les métriques définies dans la partie 5.4, car l'objectif évalué est différent.

L'évaluation est faite de la façon suivante pour quatre rôles métier à définir :

- Les propositions de permissions sont faites humainement avec la méthode opérationnelle classique. On relève le nombre de propositions humaines N_H faites pour chaque rôle.
- Les propositions humaines sont ensuite revues, et celles pertinentes pour la formation du rôle métier sont relevées. On obtient alors le nombre de permissions retenues $N_{retenues}$.
- On utilise la méthode de RM proposée pour faire des recommandations de permissions. Puisque les recommandations sont classées par score, on relève le nombre de recommandations N_{RM} à atteindre pour exprimer toutes les permissions retenues. Par exemple si on a 5 permissions retenues qui figurent respectivement dans le classement des recommandations aux places 1, 2, 4, 5 et 7, alors on relève $N_{RM} = 7$.
- On calcule ensuite la précision d'acceptation pour la méthode opérationnelle (Précision H) et la méthode de role mining (Précision RM).

On renseigne toutes ces informations dans le tableau 6.3.
On observe alors un net gain en précision avec la méthode de role mining proposée, qui est en moyenne 22% plus précise que la méthode opérationnelle. Un autre avantage de la méthode

Rôle	N_H	N_{RM}	$N_{retenus}$	**Précision H**	**Précision RM**
Rôle 1	14	12	9	64%	75%
Rôle 2	34	16	12	35%	75%
Rôle 3	14	10	10	72%	100%
Rôle 4	15	13	10	66%	77%

Table 6.3 Résultats sur la formation de rôles métier chez le partenaire industriel

proposée est le gain de temps par rapport à la méthode conventionnelle, faisant gagner environ 10 minutes pour donner l'ensemble des permissions pour former un rôle métier.

Le champ d'évaluation comparative demeure restreint, ce qui nous a conduit à ajuster le cadre d'évaluation initialement prévu. Les résultats présentés doivent donc être interprétés avec prudence. Des analyses complémentaires, appuyées par des tests supplémentaires, permettraient d'en confirmer la validité.

6.5 Discussion

L'approche proposée obtient ses meilleurs résultats sur des jeux de données tels qu'*americas_small, firewall1, firewall2*, et de grands jeux de données synthétiques sans tension utilisateur, en exprimant 90% des assignations de permissions et supprimant avec précision le bruit et le privilege creep. Sur les jeux de données du monde réel, elle réduit également le nombre de rôles requis d'un facteur de 10 dans les meilleurs cas, ce qui améliore grandement la gestion des accès dans les grandes organisations. Toutes les exécutions roulent en un temps acceptable de moins d'une minute avec une implémentation en Python, probablement améliorable en utilisant un langage compilé qui roule plus vite.

Un avantage clé de l'approche proposée est sa nature préservatrice de sécurité (security preserving) [44], puisqu'aucune nouvelle assignation de permission n'est ajoutée à l'UPA, la rendant appropriée pour l'examen des accès critiques. On confirme cette propriété lorsqu'on procède à la comparaison de la précision et du rappel utilisés pour calculer la RPL sur les figures 6.12 et 6.13. En effet, on voit que sur tous les profils de bruit et une majorité des profils sous tension, la précision descend rarement en dessous de 99%, indiquant que l'écrasante majorité des permissions retenues sont bel et bien des permissions légitimes, les rares faux positifs sont causées par des instances de bruit non supprimées. Cependant, on observe que le rappel est lui moins élevé, influant sur les mauvais scores de F-mesure. On a donc trop de faux négatifs qu'on pourrait corriger par fine tuning de la méthode.

Il convient également de noter que bien que le privilege creep de type I soit détecté avec

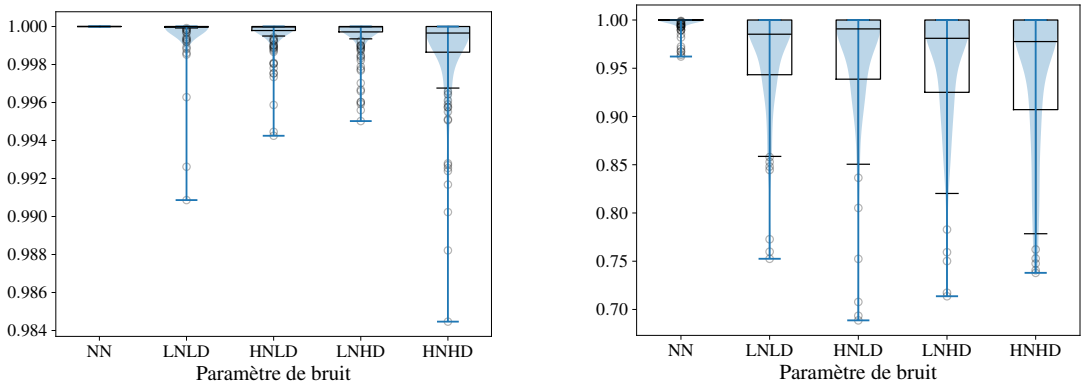
une précision supérieure à 90%, les instances de type II tendent à être plus difficiles à détecter à mesure que les niveaux de bruit augmentent, avec des scores de détection qui descendent en dessous de 50%. Cela se produit lorsque des niveaux de bruit plus élevés rendent les assignations de permissions additionnelles causées par le privilege creep de type II indiscernables du bruit aléatoire. En effet, lorsque la densité de bruit surpasse la densité d'assignations de privilege creep de type II, on observe une diminution significative de la détection de ce type de privilege creep.

En raison du processus d'analyse statistique utilisé pour nettoyer le jeu de données, les permissions issues de privilege creep de type II sont quand même effacées dans la plupart des cas, même si elles ne sont pas détectées. Ce ne serait pas le cas si le seuil t_c venait à être trop bas, auquel cas, on ajouterait du bruit à la matrice nettoyée. Bien qu'on remplisse l'objectif de nettoyage, une limitation dans la détection des instances de privilege creep liées aux projets subsiste.

Les évaluations de performance soulignent une limitation inhérente : l'approche démontre une efficacité réduite sur les jeux de données avec peu d'utilisateurs (c'est-à-dire <100) comme sur *domino, emea, healthcare*, et les jeux de données synthétiques avec tension utilisateur. En effet, la PDPC est en moyenne de 60%, et les permissions légitimes commencent à être traitées comme du bruit et sont donc supprimées.

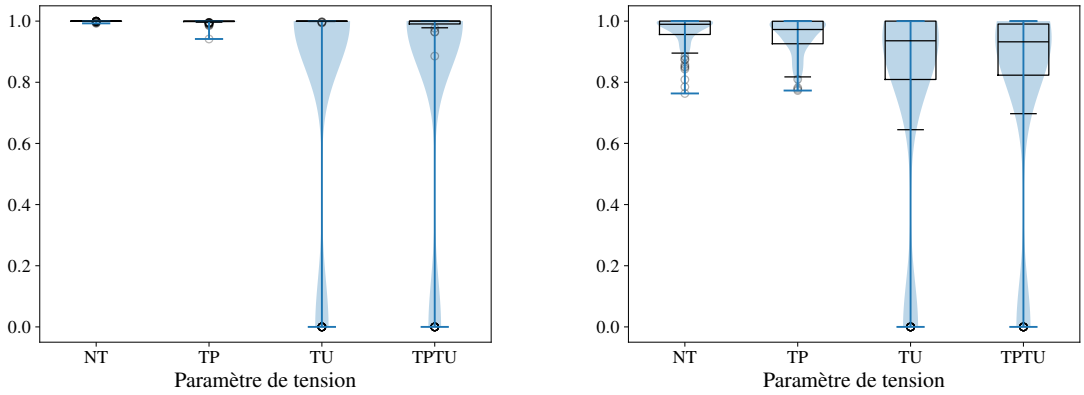
D'autres limitations proviennent des jeux de données produits synthétiquement, qui sont intrinsèquement plus propres que ceux du monde réel, et présentent des motifs structurels artificiels qui peuvent ne pas refléter la complexité et les irrégularités des environnements d'entreprise réels.

La dépendance de la méthode de nettoyage à l'ajustement des hyperparamètres est une autre limitation, même si des estimations proches sont automatiquement produites. Cette sensibilité est particulièrement importante pour le seuil de nettoyage statique t_c : la même valeur peut produire différents résultats de nettoyage selon la taille du cluster.



(a) Précision RPL (b) Rappel RPL

Figure 6.12 Comparaison entre la Précision et le Rappel pour la RPL en fonction du bruit



(a) Précision RPL (b) Rappel RPL

Figure 6.13 Comparaison entre la Précision et le Rappel pour la RPL en fonction de la tension

CHAPITRE 7 CONCLUSION

7.1 Synthèse des travaux

Ma recherche aborde la question du role mining en présence de privilege creep à travers trois contributions principales :

- Le développement d'une méthodologie de role mining efficace qui nettoie le bruit et détecte les instances de privilege creep dans les matrices UPA, atteignant 90% de précision de détection et 95% de rétention des permissions légitimes en moyenne. Cette méthode est non supervisée et n'utilise pas d'attributs la rendant agnostique aux domaines d'application et facilement déployable dans différents environnements d'entreprise sans nécessiter d'expertise spécialisée ou de données d'entraînement préalables. Elle fournit aussi des recommandations pour la revue d'accès en indiquant les utilisateurs plus probablement atteints de privilege creep.
- La construction d'un générateur de données synthétiques paramétrable qui vise à créer des jeux de données de role mining réalistes, avec injection contrôlée de bruit et d'accumulation des privilèges. Les paramètres utilisés dans le générateur sont facilement dérivables pour générer un jeu de données semblable à une structure d'entreprise donnée.
- La proposition d'un nouveau cadre d'évaluation fournissant de nouvelles métriques pour le role mining sensible au privilege creep. Ce nouveau cadre est par la suite utilisé pour évaluer la méthode de role mining proposée dans ma recherche sur des jeux de données synthétiques et réels.

L'ensemble de ma recherche a été compilée dans un article de conférence soumis et accepté à 2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2025), pour le workshop Data Security & Privacy (Data S&P). Cet article est renseigné en annexe A. L'implémentation complète des algorithmes utilisés dans cette recherche est disponible à l'adresse suivante : https://github.com/nymphargus/privilege-creep-aware-role-mining.git.

7.2 Limitations

On peut séparer les différentes limitations entre la génération de données synthétique, l'approche de role mining et le cadre d'évaluation.

Pour ce qui est de la génération de données synthétiques, la limitation principale est le fait que les jeux de données générés ne correspondent pas exactement à la réalité, et bien qu'on a essayé de rendre les jeux plus réalistes, beaucoup d'hypothèses ont été faites :

1. Les distributions des utilisateurs et des permissions sur l'arbre légitime généré peuvent ne pas suivre une loi binomiale et constituent donc une hypothèse forte sur la structure des données synthétiques.

2. Les scénarios de privilege creep et la façon dont ils se traduisent dans la matrice UPA sont limitées, d'autres scénarios pourraient être rajoutées.

3. Les jeux de données synthétiques générés sont plus propres que ceux du monde réel. Même si nous avons explicitement ajouté du bruit et du privilege creep, les motifs de permissions produits sont plus faciles à détecter que sur des jeux réels.

Cependant, on a bien validé les jeux de données générées vis-à-vis des distributions de permissions et utilisateurs générales, ces limitations sont donc nuancées.

Pour ce qui est de la méthode de role mining à proprement parler, les limitations principales viennent de la flexibilité donnée à la conception de la méthode. En effet, l'approche est sensible aux paramètres utilisés tout au long du processus (entre autre : r pour TSVD, ϵ pour DBSCAN, t_c pour le nettoyage statistique des clusters). C'est un choix voulu afin de permettre des ajustements par fine tuning. Bien qu'il existe des heuristiques assez précises pour r et ϵ, on ne dispose pas d'une telle aide pour choisir le paramètre t_c.

Cette limitation est d'autant plus flagrante lorsqu'on regarde les jeux de données réels comme *apj, customer, domino,* et *emea* qui requièrent des seuils t_c particulièrement bas pour obtenir une expression de permission d'au moins 50%. Ce phénomène provient du couplage existant entre ϵ et t_c : En essayant d'obtenir environ 10% d'outliers sur les jeux de données réels, ceci enfle les valeurs de ϵ, ce qui a comme conséquence de fusionner certains clusters. Ces clusters plus gros requièrent alors des seuils de nettoyage t_c plus faibles afin maintenir un nettoyage adéquat. Ces observations révèlent un effet cascade sur les hyperparamètres, avec ϵ et particulièrement t_c qui montrent la plus haute sensibilité. Des visualisations de clusters avec des outils comme t-distributed stochastic neighbor embedding permettraient de guider la sélection des hyperparamètres.

Enfin la plus grande limitation du processus d'évaluation est la réduction à certains profils de structure de données, de niveaux de bruit, privilege creep et tension. En effet, on a dû former des profils types suffisamment diversifiés pour évaluer la méthode à cause du nombre importants de paramètres utilisés pour la génération des données synthétiques.

7.3 Améliorations futures

Les améliorations les plus directes de l'approche proposée incluent :

- l'exploration d'algorithmes de clustering alternatifs tels qu'Isolation Forest (IF) ou Local Outlier Factor (LOF) pour le nettoyage des données permettrait d'améliorer la performance sur les jeux de données de taille restreinte.
- Le développement de seuils t_c adaptatifs basés sur les caractéristiques des clusters (par exemple un seuil plus petit à mesure qu'un cluster est grand), pourrait être une solution pour pallier la sensibilité particulière du paramètre t_c.
- L'utilisation d'attributs utilisateur pour aider l'effort de clustering en s'inspirant de l'interprétabilité de Kang et al. (voir section 2.5.1) afin de mieux guider le clustering et la sélection de rôles en aval.
- L'enrichissement des jeux de données synthétiques, avec des modèles d'architectures matricielles ou en réseau, en ajoutant des contraintes de separation of duties, permettrait de modéliser des scénarios plus proches de la réalité en entreprise.
- Une évaluation comparative des performances de la méthode proposée avec des méthodes de l'état de l'art adaptées complèterait la validation relative aux techniques existantes.

RÉFÉRENCES

[1] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber et R. E. Tarjan, "Fast exact and heuristic methods for role minimization problems," dans *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2008, p. 1–10.

[2] S. J. Stolfo, S. M. Bellovin, S. Hershkop, A. D. Keromytis, S. Sinclair et S. W. Smith, *Insider attack and cyber security: beyond the hacker*. Springer Science & Business Media, 2008, vol. 39.

[3] Fortinet. (2024) What is an insider threat? [En ligne]. Disponible: https://www.fortinet.com/resources/cyberglossary/insider-threats

[4] Cybersecurity and Infrastructure Security Agency. (2024) Defining insider threats. [En ligne]. Disponible: https://www.cisa.gov/topics/physical-security/insider-threat-mitigation/defining-insider-threats

[5] Ponemon Institute. (2023) 2023 cost of insider threats global report. [En ligne]. Disponible: https://www2.dtexsystems.com/l/464342/2023-09-15/3w7l7k/464342/1694800570ZwvyrzsD/2023_Cost_of_Insider_Risks_Global_Report____Ponemon_and_DTEX____Dgtl.pdf

[6] Cybersecurity Insiders, Securonix. (2024) 2024 insider threat report. [En ligne]. Disponible: https://www.cybersecurity-insiders.com/2024-insider-threat-report-trends-challenges-and-solutions/

[7] Capital One Data Breach Settlement Administrator. (2022) Notice of class action settlement - in re: Capital one inc. customer data security breach litigation, mdl no. 1:19-md-2915. [En ligne]. Disponible: https://www.capitalonesettlement.com/Content/Documents/Notice.pdf

[8] Office of the Comptroller of the Currency. (2020) Occ fines capital one $80 million for 2019 cyber breach. [En ligne]. Disponible: https://www.occ.gov/news-issuances/news-releases/2020/nr-occ-2020-101.html

[9] United States District Court for the Eastern District of Virginia. (2021) Order Granting Final Approval of Class Action Settlement. [En ligne]. Disponible: https://www.capitalonesettlement.com/Content/Documents/Final%20Approval%20Order.pdf

[10] Guidehouse Inc. (2023) State of maine office of the attorney general - data breach notification - tesla. [En ligne]. Disponible: https://www.maine.gov/agviewer/content/ag/985235c7-cb95-4be2-8792-a1252b4f8318/014ae6db-4cb7-464b-b827-5d73f0bbc911.shtml

[11] Handelsblatt. (2023) Digitales Dossier: Das sind die Handelsblatt-Recherchen zu den Tesla-Files. [En ligne]. Disponible: https://www.handelsblatt.com/unternehmen/industrie/digitales-dossier-das-sind-die-handelsblatt-recherchen-zu-den-tesla-files/29170110.html

[12] United States District Court for the Northern District of California. (2023) PAI v. TESLA, INC. [En ligne]. Disponible: https://www.classaction.org/media/pai-v-tesla-inc.pdf

[13] Y. A. Marquis, "From theory to practice: Implementing effective role-based access control strategies to mitigate insider risks in diverse organizational contexts," *Journal of Engineering Research and Reports*, vol. 26, n°. 5, p. 138–154, 2024.

[14] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn et R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security*, vol. 4, n°. 3, p. 224–274, 2001.

[15] Contributeurs de Wikipédia. (2024) Fonction multivaluée. Wikipédia, l'encyclopédie libre. [En ligne]. Disponible: https://fr.wikipedia.org/wiki/Fonction_multivalu%C3%A9e

[16] D. Zhang, K. Ramamohanarao et T. Ebringer, "Role engineering using graph optimisation," dans *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2007, p. 139–144.

[17] (2022) Graphe biparti. Wikipédia. [En ligne]. Disponible: https://fr.wikipedia.org/wiki/Graphe_biparti

[18] J. Vaidya, V. Atluri, Q. Guo et H. Lu, "Edge-RMP: Minimizing administrative assignments for role-based access control," *Journal of Computer Security*, vol. 17, n°. 2, p. 211–235, 2009.

[19] J. Vaidya, V. Atluri et J. Warner, "RoleMiner: Mining roles using subset enumeration," dans *Proceedings of the 13th ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2006, p. 144–153.

[20] Q. Guo, J. Vaidya et V. Atluri, "The role hierarchy mining problem: Discovery of optimal role hierarchies," dans *2008 annual computer security applications conference (ACSAC)*. IEEE, 2008, p. 237–246.

[21] J. Schlegelmilch et U. Steffens, “Role mining with ORCA,” dans *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2005, p. 168–176.

[22] J. Vaidya, V. Atluri et Q. Guo, “The role mining problem: Finding a minimal descriptive set of roles,” dans *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2007, p. 175–184.

[23] J. Vaidya, V. Atluri, Q. Guo et H. Lu, “Role mining in the presence of noise,” dans *Data and Applications Security and Privacy XXIV - 24th Annual IFIP WG 11.3 Working Conference, Proceedings*, vol. 6166 LNCS. Springer Verlag, 2010, p. 97–112.

[24] I. Molloy, N. Li, Y. A. Qi, J. Lobo et L. Dickens, “Mining roles with noisy data,” dans *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 2010, p. 45–54.

[25] M. Frank, D. Basin et J. M. Buhmann, “A class of probabilistic models for role engineering,” dans *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, p. 299–310.

[26] M. Frank, J. M. Buhman et D. Basin, “Role Mining with Probabilistic Models,” *ACM Transactions on Information and System Security*, vol. 15, n°. 4, p. 15:1–15:28, 2013.

[27] A. Kern, A. Schaad et J. Moffett, “An administration concept for the enterprise role-based access control model,” dans *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2003, p. 3–11.

[28] A. P. Streich, M. Frank, D. Basin et J. M. Buhmann, “Multi-assignment clustering for Boolean data,” dans *Proceedings of the 26th Annual International Conference on Machine Learning*. Association for Computing Machinery, 2009, p. 969–976.

[29] M. Frank, A. P. Streich, D. Basin et J. M. Buhmann, “Multi-assignment clustering for boolean data,” *J. Mach. Learn. Res.*, vol. 13, n°. 1, p. 459–489, 2012.

[30] Z. Ghahramani et T. Griffiths, “Infinite latent feature models and the indian buffet process,” *Advances in neural information processing systems*, vol. 18, 2005.

[31] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das et H. Mannila, “The discrete basis problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, n°. 10, p. 1348–1362, 2008.

[32] A. Kabán et E. Bingham, “Factorisation and denoising of 0–1 data: a variational approach,” *Neurocomputing*, vol. 71, nº. 10-12, p. 2291–2308, 2008.

[33] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang et J. Lobo, “Evaluating role mining algorithms,” dans *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2009, p. 95–104.

[34] H. Huang, F. Shang, J. Liu et H. Du, “Handling least privilege problem and role mining in RBAC,” *Journal of Combinatorial Optimization*, vol. 30, nº. 1, p. 63–86, 2015.

[35] H. Lu, Y. Hong, Y. Yang, L. Duan et N. Badar, “Towards user-oriented RBAC model,” *Journal of Computer Security*, vol. 23, nº. 1, p. 107–129, 2015.

[36] B. Mitra, S. Sural, J. Vaidya et V. Atluri, “A Survey of Role Mining,” *ACM Computing Surveys*, vol. 48, nº. 4, p. 50:1–50:37, 2016.

[37] J. Jiang, X. Yuan et R. Mao, “Research on Role Mining Algorithms in RBAC,” dans *Proceedings of the 2018 2nd High Performance Computing and Cluster Technologies Conference*. Association for Computing Machinery, 2018, p. 1–5.

[38] M. Trnecka et M. Trneckova, “An incremental algorithm for the role mining problem,” *Computers & Security*, vol. 94, p. 101830, 2020.

[39] C. Blundo, S. Cimato et L. Siniscalchi, “Role Mining Heuristics for Permission-Role-Usage Cardinality Constraints,” *Computer Journal*, vol. 65, nº. 6, p. 1386–1411, 2022.

[40] Q. Guo et M. Tripunitara, “The Secrecy Resilience of Access Control Policies and Its Application to Role Mining,” dans *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2022, p. 115–126.

[41] L. Dong, T. Wu, W. Jia, B. Jiang et X. Li, “Computable Access Control: Embedding Access Control Rules Into Euclidean Space,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, nº. 10, p. 6530–6541, 2023.

[42] C. Blundo et S. Cimato, “Role mining under User-Distribution cardinality constraint,” *Journal of Information Security and Applications*, vol. 78, p. 103611, 2023.

[43] F. Zhu, C. Yang, L. Zhu et J. Gu, “Application of Matrix Factorization Role Mining Algorithm in Role-Based Access Control for Edge RMP,” dans *2024 9th International Conference on Electronic Technology and Information Science (ICETIS)*, 2024, p. 761–767.

[44] J. Crampton, E. Eiben, G. Gutin, D. Karapetyan et D. Majumdar, “Generalized Noise Role Mining,” dans *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2022, p. 91–102.

[45] M. Abolfathi, Z. Raghebi, H. Jafarian et F. Banaei-Kashani, “A Scalable Role Mining Approach for Large Organizations,” dans *Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics*. Association for Computing Machinery, 2021, p. 45–54.

[46] S. D. Stoller, P. Yang, C. R. Ramakrishnan et M. I. Gofman, “Efficient policy analysis for administrative role based access control,” dans *Proceedings of the 14th ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2007, p. 445–455.

[47] S. Parkinson, S. Khan, J. Bray et D. Shreef, “Creeper: A tool for detecting permission creep in file system access controls,” *Cybersecurity*, vol. 2, n°. 1, p. 14, 2019.

[48] D. Alexander et D. K. Chikwarti, “Graph-Based AI Techniques for Role Mining and Access Optimization in Complex Enterprises,” *International Journal of Advanced Engineering Technologies and Innovations*, vol. 01, n°. 03, 2023.

[49] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo et J. Lobo, “Mining roles with semantic meanings,” dans *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2008, p. 21–30.

[50] H. Lu, J. Vaidya et V. Atluri, “An optimization framework for role mining,” *Journal of Computer Security*, vol. 22, n°. 1, p. 1–31, 2014.

[51] S. Vavilis, A. I. Egner, M. Petković et N. Zannone, “Role Mining with Missing Values,” dans *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, p. 167–176.

[52] S. Parkinson et A. Crampton, “Identification of irregularities and allocation suggestion of relative file system permissions,” *Journal of Information Security and Applications*, vol. 30, p. 27–39, 2016.

[53] S. D. Stoller et T. Bui, “Mining hierarchical temporal roles with multiple metrics,” *Journal of Computer Security*, vol. 26, n°. 1, p. 121–142, 2018.

[54] N. Gal-Oz, Y. Gonen et E. Gudes, “Mining meaningful and rare roles from web application usage patterns,” *Computers & Security*, vol. 82, p. 296–313, 2019.

[55] H. Kang, G. Liu, Q. Wang, Q. Zhang, J. Niu et N. Luo, "An improved minimal noise role mining algorithm based on role interpretability," *Computers & Security*, vol. 127, p. 103100, 2023.

[56] O. Durdag et A. Coskuncay, "Reconfiguring Role-Based Access Control via Role Clustering," *IEEE Access*, vol. 13, p. 72 984–72 993, 2025.

[57] M. N. Nobi, R. Krishnan, Y. Huang, M. Shakarami et R. Sandhu, "Toward Deep Learning Based Access Control," dans *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '22. Association for Computing Machinery, 2022, p. 143–154.

[58] S. Parkinson et S. Khana, "Identifying high-risk over-entitlement in access control policies using fuzzy logic," *Cybersecurity*, vol. 5, n°. 1, p. 6, 2022.

[59] M. C. Massi, F. Ieva et E. Lettieri, "Data mining application to healthcare fraud detection: A two-step unsupervised clustering method for outlier detection with administrative databases," *BMC Medical Informatics and Decision Making*, vol. 20, n°. 1, p. 160, 2020.

[60] Y. Lu, I. Cohen, X. S. Zhou et Q. Tian, "Feature selection using principal feature analysis," dans *Proceedings of the 15th ACM International Conference on Multimedia*. Association for Computing Machinery, 2007, p. 301–304.

[61] T. I. A. Souza, A. L. L. Aquino et D. G. Gomes, "A method to detect data outliers from smart urban spaces via tensor analysis," *Future Generation Computer Systems*, vol. 92, p. 290–301, 2019.

[62] G. Bergqvist et E. G. Larsson, "The Higher-Order Singular Value Decomposition: Theory and an Application [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 27, n°. 3, p. 151–154, 2010.

[63] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," dans *kdd*, vol. 96, n°. 34, 1996, p. 226–231.

[64] Wikipedia. (2025) Dbscan. [En ligne]. Disponible: https://fr.wikipedia.org/wiki/DBSCAN

[65] H. Lu, J. Vaidya et V. Atluri, "Optimal Boolean Matrix Decomposition: Application to Role Engineering," dans *2008 IEEE 24th International Conference on Data Engineering*, 2008, p. 297–306.

[66] L. Dong, T. Wu, W. Jia, B. Jiang et X. Li, "Computable Access Control: Embedding Access Control Rules Into Euclidean Space," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, nº. 10, p. 6530–6541, 2023.

ANNEXE A ARTICLE 1 : A PRIVLEGE CREEP-AWARE ROLE MINING METHOD FOR ENHANCED ACCES CONTROL SECURITY, SOUMIS À IEEE TRUSTCOM 2025 WORKSHOP DATA S&P LE 4 OCTOBRE 2025

A Privilege Creep-Aware Role Mining Method for Enhanced Access Control Security

Vincent Bittard, Rim Ben Salem, Ahmed Bouzid,
Sara Imene Boucetta, Frédéric Cuppens, Nora Cuppens-Boulahia
Department of Software and Computer Engineering, *Polytechnique Montréal*, Montréal, Canada
{vincent.bittard, rim.ben-salem, ahmed.bouzid, sara-imene.boucetta, frederic.cuppens, nora.boulahia-cuppens}@polymtl.ca

***Abstract*—Role Mining (RM) extracts Role-Based Access Control (RBAC) structures from user-permission assignments to reduce administrative overhead. However, existing approaches usually make the assumption of clean datasets, while real-world systems suffer from anomalies like privilege creep, the gradual accumulation of unnecessary permissions.**

The proposed approach aims to detect potential privilege crept users who should be reviewed first, and identify legitimate permissions assignments to be expressed in RBAC, reducing management complexity. It consists of a two-step procedure: clean the User-Permission Assignment matrix (UPA) using a clustering and statistical analysis, then build an RBAC state using a regular role mining algorithm.

The proposed approach yields an average of 90% in privilege creep detection accuracy and over 95% privilege creep correction, evaluated on synthetically made datasets. Evaluation on real-world datasets demonstrates an average 4-fold reduction in required roles while maintaining at least 80% UPA coverage.

***Index Terms*—Role mining, RBAC, access control, clustering, anomaly detection**

I. Introduction

To alleviate Identity and Access Management (IAM) costs, system administrators often structure permission assignments using RBAC, grouping users with similar need-to-know. This enables a more scalable and maintainable access control configuration. Recent approaches to role mining have focused on methods designed for clean datasets. In real-world access control systems, this assumption rarely holds. Over time, the quality of IAM systems degrades, with privilege creep representing a key anomaly in insider threat contexts: an incremental accumulation of permissions resulting from job transitions, organizational changes, temporary assignments etc. While this issue can be manually addressed in small organizations, the complexity increases significantly in larger enterprises, especially since permission review is often fragmented and delegated.

The process addressing this issue is known as Noise Role Mining [1], [2], [3], [4]. Noise definitions commonly fall under two categories : administration errors (wrongly granted or revoked permissions, regardless of cause) or applicability exceptions (legitimate but policy-complicating permissions). Therefore, noise-aware role mining techniques can be viewed as a subset of privilege creep-aware techniques, aiming to remove noisy assignments directly, with the goal to improve security or reduce management costs.

Historically, first approaches focused on mining roles directly on noisy data. Vaidya et al. [1] introduce the δ-Role Mining Problem (RMP) and Minimal Noise RMP to fulfil this goal, which allow partial UPA expression by tolerating mismatches between the source UPA and the mined one, effectively removing noise. Multi-Assignment Clustering (MAC) by Frank et al. [5] similarly allows partial expression, and clusters users with overlapping permissions, discarding low-probability assignments to remove noise.

Other approaches use a 2-step method: clean the data first and mine later. Molloy et al. [2] use binary matrix decomposition algorithms for the cleaning step. Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), Binary Non-Negative Matrix Factorization (BNMF) and logistic Principal Component Analysis (PCA) are compared to δ-RMP and MAC. The UPA is decomposed and reconstructed in binary form via inverse transformation, using a step function to enforce binary values. Since the decompositions are not exact, some data is destroyed, hence removing noise. The evaluation by Molloy et al. [2] demonstrates that the 2 step-method yields better noise-removing results on synthetic datasets than the other previously mentioned approaches.

The main contributions of this paper are:

- A new parameterizable synthetic data generator that aims to build realistic RM datasets, able to inject noise and privilege creep related to common enterprise scenarios.
- A proposed approach to clean RM datasets by removing potential noise and detecting instances of privilege creep, in order to mine roles on the cleaned dataset.
- A thorough performance evaluation introducing new metrics to assess the accuracy of the privilege creep removal.

The remainder of this paper is structured as follows: Section II reviews recent approaches relevant to the problem addressed in this paper. Afterward, Section III and IV define the basis of RM, the terminology, and the nomenclature for noise and privilege creep used in this paper. Then, the proposed privilege creep-aware role mining approach is explained in section V, while the synthetic dataset generation method is described in section VI. Next, section VII sets the evaluation metrics and benchmark for validation. Section VIII then presents results on synthetic and real-world datasets. Finally section IX provides a discussion about the advantages, shortcomings, and potential improvements of the proposed method.

II. Related work

A. Recent role mining approaches

Kang et al. [4] improve previous noise RM algorithms using role interpretability. The mined role set is given a user attribute-based interpretability score to minimize alongside reconstruction error. This produces roles that more appropriately fit business needs, ultimately used to reduce management overhead. While the issue of noise is tackled explicitly in this work, its nature remains unclear and privilege creep is not directly addressed as the algorithm goal is to approximate existing access patterns.

Durdag and Coskuncay [6] choose to reconfigure RBAC systems by clustering similar roles based on permission similarity using Agglomerative Hierarchical Clustering (AHC). Identified clusters serve as reference structures to support system redesign or cleanup, improving manageability of complex RBAC configurations. Their approach removes problematic data during preprocessing and validates against expert expectations, but lacks mechanisms to identify or correct anomalous permission patterns.

Wang et Wu [7] propose a method to reduce role proliferation in RBAC systems using formal concept analysis and concept lattice factorization. The method first generates an initial state, then optimizes by balancing user-role and permission-role assignments while reducing the concept lattice dimensionality. This allows the establishment of role mining objectives before algorithm execution. However, the authors note needs for benchmarking, scalability improvements, and noise robustness testing against existing methods.

Nobi et al. [8] introduce Deep Learning Based Access Control (DLBAC), neural networks that learn directly from raw user and resource metadata, eliminating the need for manual engineering of roles, attributes, and policies. The prototype DLBAC_α demonstrates superior accuracy and generalization compared to classical policy mining and machine learning approaches, while addressing explainability concerns through interpretation techniques like Integrated Gradients. However, DLBAC does not directly tackle privilege creep because it learns from existing authorization data that may already contain anomalous access rights. This limitation is acknowledged by the authors, stating that errors in the datasets used could introduce bias in the trained model.

The work in this paper differs from the recent approaches by explicitly modeling privilege creep scenarios separately from generic noise, and evaluating performance based on the algorithm's ability to structure legitimate permissions while identifying and removing excessive permissions.

B. Recent privilege creep detection approaches

Parkinson et al. [9] present an unsupervised tool to detect privilege creep instances in file system Access Control Lists (ACL) using χ^2 statistics, establishing an average 96% accuracy in privilege creep detection on synthetic datasets. The approach is scientifically sound but suffers from a key evaluation flaw: using accuracy instead of F-measure artificially inflates performance since legitimate users greatly outnumber anomalous instances, causing high true negative counts to skew the metric despite an average 30% false negative rate in detecting actual privilege creep. Parkinson et al. [10] also propose a fuzzy logic-based approach to identify critical privilege creep in access control policies by modeling user trust, resource sensitivity, and permission power as fuzzy sets rather than binary classifications. This new approach is reliant on security event logs, producing better results than the Creeper tool but still suffering from the same evaluation framework flaw.

Alexander and Chikwarti [11] propose a graph-based AI framework that models enterprise IAM as a knowledge graph, applying Graph Neural Networks (GNNs), community detection algorithms (Louvain method), and graph clustering to discover latent role structures from access patterns, while using Graph Autoencoders and Isolation Forests to detect privilege creep and anomalous permissions. Their approach claims to reduce role redundancy by 38% and achieve 93.5% precision in anomaly detection. However, critical limitations include evaluation exclusively on synthetic datasets, and comparison against an inappropriate baseline using k-means, unsuited for anomaly detection, rendering the reported F1-score improvement from 74.8% to 91.3% potentially misleading.

This paper diverges by introducing an unsupervised role mining approach independent of user attributes or event data, dedicated evaluation metrics, and a parameterizable synthetic generator that realistically injects privilege creep, addressing evaluation limitations and generic noise models in prior work.

III. Definitions

The National Institute of Standards and Technology (NIST) [12] formally defines the common role mining context:

- Let U, P the set of users and the set of permissions respectively
- Let $UPA \subseteq U \times P$ the binary user-permission assignment matrix, a many-to-many mapping.

The NIST also formalized the Basic Role Mining Problem (RMP) [12] as a binary matrix decomposition problem:
Given the common role mining context, find a set of roles R and two binary matrices $UA \subseteq U \times R$, the user-to-role assignment matrix and $PA \subseteq P \times R$, the permission-to-role assignment matrix where $UPA = UA \times PA$ minimizing $|R|$.

In this paper, a permission refers to an existing column in the UPA matrix, a user refers to an existing row in the UPA matrix, and a permission assignment is defined as an existing mapping in the UPA matrix. The concept of legitimate permission is defined as permission assignments that should be mined during the RM process, as opposed to noise or privilege creep that should not be mined.

IV. Identifying noise and privilege creep

A. Noise

The proposed nomenclature draws inspiration on the types of noise identified by Molloy et al. [2] and Vaidya et al. [1]:

- Correctness noise: isolated administration errors that usually occur when a user goes through access provisioning for the first time, only affecting a small set of permissions. Correctness noise is usually additive, meaning additional permission assignments on affected users, because of the availability issue subtractive noise causes.
- RBAC applicability noise: legitimate permissions that are not sufficiently shared amongst users to be usefully expressed into RBAC. Indeed, expressing them would increase the complexity of the RBAC state and therefore undermine the management advantages of maintaining this structure for access control.

B. Privilege creep

Two privilege creep scenarios are considered:

- Scenario 1: an employee moves to a different position within the organization but retains a subset of permission assignments from their previous duties due to an incomplete deprovisioning. Let's call this type I privilege creep.
- Scenario 2: a group of employees was assigned to a now finished temporary project. The additional permissions they were given to fulfil their responsibilities have not been revoked entirely. This can occur due to unmaintained records of projects. Let's call this type II privilege creep.

The key difference between these two privilege creep types is the subset of permissions impacted.

1) Type I: The impacted permissions are still legitimate for other employees. Indeed, those currently in the same position as the privilege crept employee previously held, or the employee who takes the new vacant position, have a valid need for these permissions to fulfil their task.

2) Type II: The impacted permissions are not legitimate for employees outside the project. Indeed, the permissions are project-bound and once the project ends, no employee should retain them.

V. Privilege creep-aware role mining

As explained in section I, the proposed method uses a 2-step approach : clean the dataset and then mine the roles with a regular role mining algorithm.

A. Prerequisite

Translate the user-permission assignments into a standard binary UPA matrix X as defined in section III.

B. Dimensionality reduction

The first step is to use a dimensionality reduction algorithm. This is done using Truncated-SVD (TSVD). TSVD offers faster computing times on sparse matrices, does not require centered data, and its partial decomposition of X reduces the impact of noise [2]. Indeed, the assumption that UPA matrices are sparse usually holds true [1] [2] [13] [3].

Let $k \in \mathbb{N}^*$ the rank of the TSVD decomposition of X, where k represents the dimensionality of the reduced space. The low-dimensional embedding Z of X is obtained by the following equation :

$$Z = XV_k^T \tag{1}$$

Where V_k is the top k right singular vectors matrix of X.

For each component $i \in [\![1, k]\!]$, Z_i the i-th column of Z. The empirical variance of the i-th component corresponds to the variance of Z_i :

$$Var(Z_i) = \frac{1}{n}\sum_{j=1}^{n}(Z_{ji} - \mu_i)^2 \text{, where } \mu_i = \frac{1}{n}\sum_{j=1}^{n} Z_{ji} \tag{2}$$

The variance of X is computed with:

$$Var(X) = \sum_{j=1}^{d} Var(X_j) = \sum_{j=1}^{d} p_j(1 - p_j) \tag{3}$$

Where $p_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij}$ is the fraction of ones in column j. The total explained variance ratio R_k is computed using the explained variance ratio r_i for each component $i \in [\![1, k]\!]$ using the following equation:

$$R_k = \sum_{1}^{k} r_i = \sum_{1}^{k} \frac{Var(Z_i)}{Var(X)} \tag{4}$$

The goal is to find a TSVD rank k that renders around 80% total explained variance ratio [2]. Since this sole condition can produce an overwhelmingly large number of components k for the next step in the approach, another condition is added on r_k. The search stops when the following condition is met :

$$R_k > 0.80 \text{ OR } r_k < \epsilon \tag{5}$$

At this point, adding more and more components yields diminishing returns. Empirically, setting $\epsilon = 0.02$ provides reliable results.

C. Clustering and outlier identification

Z should contain less noise than X and consists of real valued components that capture the most important user-permission relationships. Hence, clustering is performed on the low-dimensional embedding Z.

Since the primary objective of the proposed method is to detect privilege creep, choosing a clustering algorithm capable of identifying anomalous users as outliers becomes necessary. The selected clustering algorithm to fulfil this goal is Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which offers easily interpretable parameters to be determined :

- min_{points} the minimum number of points required to form a cluster. Since clusters should identify users who present similar permissions, which usually occurs in teams of employees working together, min_{points} is set to the smallest team size detectable in the UPA.
- ϵ the radius of the neighborhood with respect to some point. To determine an ideal value for ϵ the 1 Nearest-Neighbor (1-NN) Euclidean distance of all users are

computed and plotted in ascending order. Then, ϵ is set to the cutoff value corresponding to the elbow of the curve. In the proposed approach, the elbow is detected automatically using the normalized difference curve technique. The ϵ value can still be overwritten manually using the visual method.

Let's call C, the user-cluster assignment table giving the DBSCAN labels for each user including the outliers.

D. Cleaning and role mining

In order to clean the UPA C and X are used together by performing a statistical analysis on each cluster to determine which permissions assignments are to be expressed. Outliers are first removed off of X to ensure role quality. Permission prevalences are then computed for each cluster, defined as the share of users within a cluster who possess a given permission. Using a threshold t_c the permissions with prevalence less than t_c are removed and the permissions with prevalence greater than t_c are kept. The matrix obtained after the cleaning process is called the cleaned UPA K.

E. Role mining

FastMiner [14] and Optimal Boolean Matrix Decomposition using BasicRMP introduced by Lu et al. [15] are used as the role mining algorithm with a greedy approach. The algorithm is run on K with the outliers removed. It expresses the assignments entirely without approximations. After the role mining is done, outliers need to be reintroduced into K.

F. Reunification

This step is a new addition from previous methods in the literature that do not identify outliers. Two use cases are considered:

1) Omniscient reunification: This is the case used for synthetic datasets, adding the outliers back into the cleaned UPA K with their known legitimate permissions. This emulates the presence of an expert who is able to clean the privilege creep and noise out of a user permission pool perfectly.
2) Heuristic reunification: This is the case in which the proposed tool will most likely be used in an organization setting to provide recommendations for outliers. Two concurrent approaches are used. The first one is to assign mined roles to the outliers even though all their permissions may not be covered. This gives a rough idea of potential roles this user could have, and also brings out the potentially problematic permissions that are not covered. The second approach uses the information on clusters: The barycenter of every cluster using the binary coordinates of users is computed. Then, every outlier is linked to the closest barycenter. This gives additional insight on which cluster the identified outlier could be part of and their potential legitimate permissions. Ultimately the decision should be made by an administrator in charge of reviewing user accounts as both approaches could provide conflicting information.

The proposed approach is evaluated using the omniscient reunification. Thus, the metrics chosen in section VII-C account for the accuracy of outlier detection. The following section details the generation of synthetic datasets, which is another contribution that this paper brings forth.

VI. Synthetic datasets

Given that the proposed approach is evaluated on privilege creep detection and noise correction, and that existing synthetic dataset generators in the literature do not account for privilege creep, a new dataset generation method is required. One of the main contributions of this paper is a flexible generator with adjustable parameters to produce synthetic datasets mimicking real-world ones.

A. Generating legitimate permissions

To build the generator, inspiration was taken from the Tree-Based Data Generator from Molloy et al. [16]. The main changes are made on the propagation strategy to generate trees, the process of adding noise, and the new process of adding privilege creep instances. The proposed generator produces a mock business organization hierarchy structure using a tree.

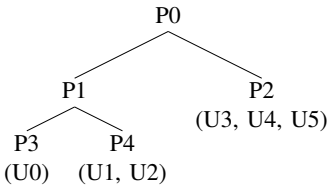


Figure 1: Template example; U1,U2 inherit permission sets P0, P1 and P4, U0 inherits permission sets P0, P1, P3 etc.

The tree structure, assigned with permissions and users, is referred to as the template. As a RM objective, mined roles should correspond to the template structure. Knowing this, templates are used to create legitimate permission hierarchies.

The generator starts with the root node and then grows child nodes iteratively, using six parameters to build trees:

- min_depth and max_depth, encode the minimum and maximum depth of the tree, nodes stop propagating between these bounds with a linearly decaying probability.
- min_children and max_children encode the minimum and maximum number of child nodes allowed per node.
- avg_branch and std_dev encode how nodes propagate in the tree. The number of child nodes follows a normal distribution centered around avg_branch with a standard deviation of std_dev, brought back to integers.

Given a target number of permissions and users, a random number of permissions is assigned to every node using a binomial law. A random number of users is assigned to leaves of the tree using the same binomial law method. Users then inherit the permissions of the leaf node they are assigned to, and all permissions assigned to parents nodes above them (see example on figure 1). This process generates the legitimate UPA matrix.

B. Adding privilege creep

As identified in section IV, two types of privilege creep are added to the UPA matrix: type I and type II privilege creep. Three parameters are introduced for this purpose:

1) p_{PC} the portion of users who present cases of type I privilege creep.
2) c the portion of permission assignments copied from another user in a type I privilege creep instance.
3) r the number of added permissions for users who present type II privilege creep. All affected users are then granted these permissions.

In order to always have a varied set of privilege creep instances the following rules are used:

- 30% of type I privilege creep instances use $c = 1$
- 70% of type I privilege creep instances are assigned linearly decreasing values of c from 1 to 0.
- The number of users in each type II privilege creep instance is fixed to 8, and the number of instances is computed using this formula:

$$n_{\text{Type II instances}} = \lfloor \log_{10}(n_{users}) \rfloor \tag{6}$$

Thanks to these parameters, it is possible to produce privilege creep instances of varying frequency and magnitude. The privilege crept UPA is then created by adding the privilege crept permissions assignments to the legitimate UPA.

C. Adding noise

To make the dataset more realistic, noise is added to the privilege crept UPA. This follows the discussion in section IV on noise identification. RBAC applicability noise is added using the first two parameters and correctness noise using the third parameter :

- p_{noise}: noise percentage denotes the ratio of noisy permission assignments to be added to the UPA matrix, expressed as a proportion of the number of legitimate assignments.
- d_{noise}: noise density denotes how dense the added assignments are. The number of distinct added permissions N_{noisy} must be computed to match the noise density parameter. This is done using the formula:

$$N_{noisy} = \frac{N_{legit} p_{noise}}{d_{noise} n_{users}} \tag{7}$$

  Where N_{legit} is the number of legitimate assignments. The added noise assignments are then generated with a Bernoulli experience on a matrix of size (n_{users}, N_{noisy}) with $p = d_{noise}$, concatenated to the privilege crept UPA.
- $p_{legit-noise}$, noise percentage on legitimate permissions indicates the ratio of additional assignments to introduce relative to the number of already injected noisy permissions. These assignments are uniformly distributed on the legitimate permission matrix directly, only flipping zeros into ones.

VII. Evaluation

The synthetic dataset evaluation framework is first defined. Then, the real-world datasets used for evaluation are mentioned. Finally, Metrics are listed at the end of the section.

A. Synthetic dataset benchmark

Since it is impossible to test every parameter combination with the proposed dataset generation method, standard configurations are defined for a variety of problem sizes, general hierarchy aspect of the legitimate templates, and privilege creep & noise distributions using profiles:

- Table I defines profiles used to build template trees (section VI-A), selected to have a variety of depth and branching, mimicking different organizational structures.
- Table II defines profiles used to add noise and privilege creep to the legitimate UPA (section VI-C), selected based on the assumption that the number of privilege creep instances increases with higher noise levels.
- Table III defines profiles used to assign users and permissions to tree nodes (section VI-A), selected to restrict the number of permissions assigned to nodes and the number of users assigned to leaves.

Evaluation on noise levels (2), spans across the parameter profiles from tables I and II, using the default tension parameters from table III. Evaluation under tension (3), spans across the parameter profiles from tables I and III, using the default noise parameters from table II.

Name	children		depth		avg_branch	std_dev
	min	max	min	max		
large_flat	2	4	2	4	3	1.5
small_flat	1	5	1	3	3	1
large_string	1	2	10	15	1.7	0.5
small_string	1	2	5	8	1.6	0.4
binary_tree	1	3	2	5	2	0
highly_random	1	6	2	5	2	2

Table I: Profiles for generating legitimate permission trees

Name	Acronym	Noise			
		percent	density	legit	PC
no noise no pc	NN	0	N/A	0%	0%
low noise, low density	LNLD	5%	1%	10%	3%
high noise, low density	HNLD	15%	1%	15%	5%
low noise, high density	LNHD	5%	4%	15%	5%
high noise, high density	HNHD	15%	5%	20%	8%
default	/	15%	2%	10%	3%

Table II: Profiles for noise and privilege creep levels

Name	Acronym	User		Permission	
		min	max	min	max
no tension	NT	15	25	10	40
tension on permissions	TP	15	25	2	6
tension on users	TU	2	8	10	40
tension on both	TUTP	2	8	2	6
default	/	15	25	15	45

Table III: Profiles for tension on users and permissions

B. Real-world dataset benchmark

The real-world datasets provided by Ene et al. [17] are used for the evaluation. Despite their publication in 2008, these datasets have become established benchmarks in the

literature and continue to be referenced in recent studies for performance comparison on real-world instances [4], [18], [19], [20], [21]. These are: *americas_large, americas_small, apj, customer, domino, emea, firewall1, firewall2, healthcare*.

C. Metrics

The main goal of the chosen metrics is to measure how accurate the privilege creep detection and correction is without compromising permissions flagged as legitimate.

1) Legitimate Permission Retention (LPR):
   The F1-score computed between the cleaned UPA and the reference legitimate UPA, excluding potential outliers. It reflects the accuracy of legitimate permission recovery, while minimizing the retention of noise.
2) Permission Expression (PE):
   The number of permission assignments retained in the cleaned UPA divided by the number of assignments in the noised UPA. An effective cleaning process would result in a permission expression value that is close to the one computed on the legitimate matrix.
3) Privilege Creep Correction (PCC):
   The percentage of privilege crept assignments still present in the cleaned UPA. A value close to 1 reflects the effectiveness of the cleaning process.
4) Privilege Creep Detection Accuracy (PCDA):
   The F1-score computed between the set of identified outliers and the ground truth set of anomalous users. It measures the accuracy of the anomaly detection.
5) Role Count Deviation (RCD):
   The difference between the number of ideal roles, identified to be the number of leaves in the template (see section VI-A) and the number of mined roles divided by the number of ideal roles.
6) Runtime metric:
   The runtime of the cleaning process in seconds.

On synthetic datasets, the proportion of identified privilege creep instances is also recorded by type.

VIII. Results

min_{points} is set to 5 to ensure clusters represent meaningful user groups (teams/departments) as explained in section V-C. Empirical testing has shown that lower values produce fragmented clusters where the statistical threshold t_c incorrectly removes legitimate permissions, as small clusters exhibit high variance in permission prevalence.

A. Performance using synthetic datasets

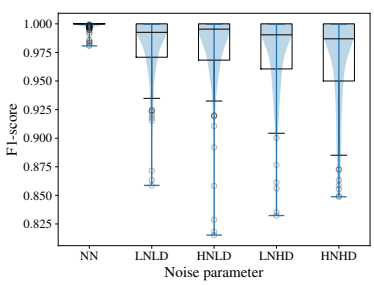
The results are given as box plots. Each plot renders the distribution of a given metric evaluated on 20 runs of each structure parameter defined on table I. Given that there are 6 structure parameters, each box plot contains 120 different samples. Metrics are explained in section VII-C.

1) Experiment with varying noise levels: Figures 2a and 2b concurrently show that legitimate permission assignments were accurately retrieved, avoiding the expression of noisy assignments. Indeed, the median LPR is above 97.5% on all noise levels and permission expression shows a proportional decrease with increasing noise levels, as expected. About privilege creep, conflicting information seem to stem from figures 2c and 2d: the detection accuracy hovers around 80%, sometimes reaching values below 50% on the worst runs, while almost all privilege crept assignments are corrected with median PCC values at 100% across all noise levels. This occurs as type I privilege creep instances are accurately detected most of the time, whereas type II instances are more difficult to detect. The statistical analysis cleaning process then removes undetected type II privilege creep permission assignments. Figure 2e reveals that the proposed approach tends to produce fewer roles than expected VI-A, probably due to legitimate permissions being erased by the cleaning process. The average runtime for the cleaning process is below 5 seconds, never exceeding 1 minute as figure 2f demonstrates. Longer runtimes can be attributed to exceptionally large datasets. When the datasets are free from noise and privilege creep, figures 2b, 2a and 2e confirm that the approach accurately retrieves legitimate assignments with minimal false positive outlier occurrences, and produces the expected amount of roles on average.

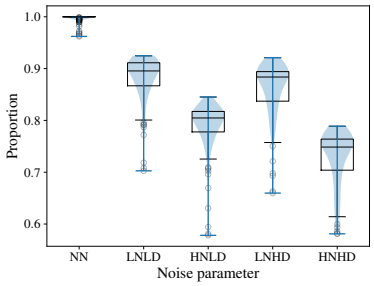
2) Experiment with varying tension levels: Figures 3a and 3b demonstrate that legitimate permissions assignments are generally not accurately retrieved when tension is put on users. Indeed, LPR plummets to 0% on several of these datasets and PE is lower than expected, reaching values that do not correspond to the amount of noise, meaning legitimate assignments have been removed massively. Regarding privilege creep, detection accuracy decreases by 30% when tension is applied to users, with many false positives as shown in figure 3d. All of this can be explained when too few users are present in the UPA matrix to form large significant groups: the cleaning process removes larger amounts of permissions due to wrongly identified clusters, which also causes a lot of users to be wrongly flagged as outliers. These effects propagate to the role mining step, no roles can be mined when great amounts of permissions have been removed. This is shown in figure 3e where runs under user tension produce half the roles that would normally be needed to express the assignments on average. Cleaning runtimes are faster than the previous experiments, averaging 3 seconds due to the datasets reduced size under user and/or permission tension.

B. Performance using real-world datasets

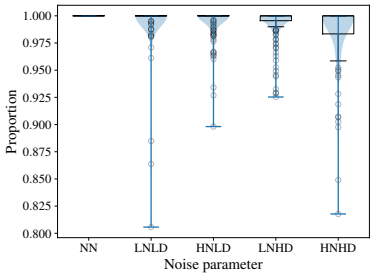
On real-world datasets, hyperparameters are determined through a semi-automated process combining the algorithmic methods from sections V-B and V-C with manual refinement. The automated procedures provide initial recommendations for n_{comps}, ϵ, and t_c. These initial values are then manually adjusted based on the following criteria: (1) maximizing the explained variance ratio R_{var} while respecting the diminishing



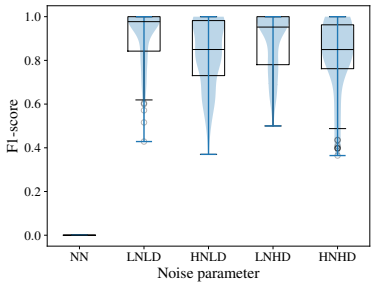
(a) Legitimate Permission Retention (LPR)



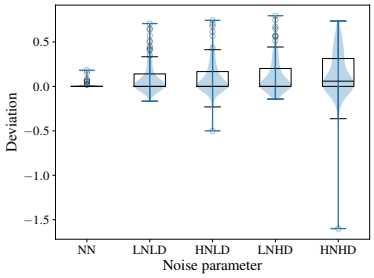
(b) Permission Expression (PE)



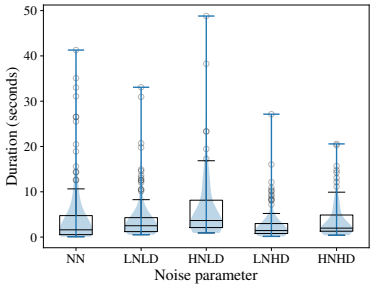
(c) Privilege Creep Correction (PCC)



(d) Privilege Creep Detection Accuracy (PCDA)

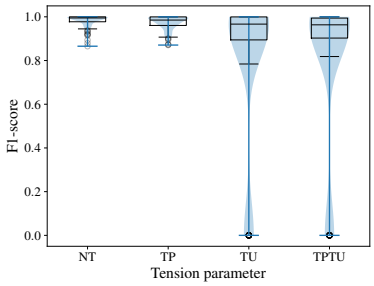


(e) Role Count Deviation (RCD)

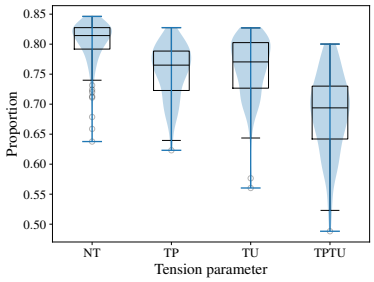


(f) Runtime

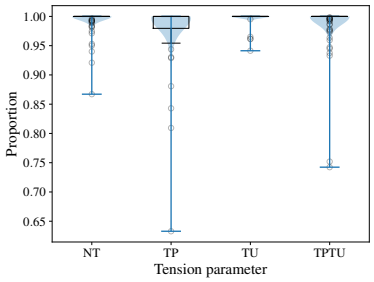
Figure 2: Performance analysis against noise levels



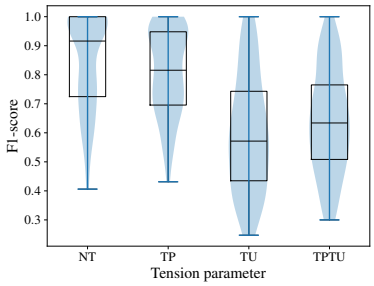
(a) Legitimate Permission Retention (LPR)



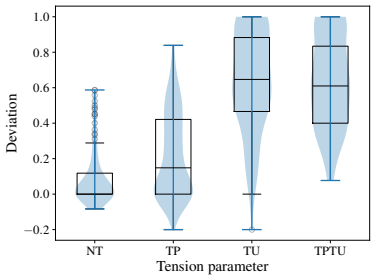
(b) Permission Expression (PE)



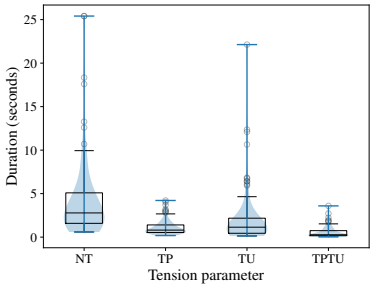
(c) Privilege Creep Correction (PCC)



(d) Privilege Creep Detection Accuracy (PCDA)



(e) Role Count Deviation (RCD)



(f) Runtime

Figure 3: Performance analysis against tension levels

returns threshold described in section V-B, (2) ensuring stable cluster formation that effectively separates users with irregular permissions patterns from the others using T-distributed stochastic neighbor embedding (t-SNE) projections, and (3) optimizing the trade-off between permission expression coverage and noise removal through t_c calibration, guided by the established 80-20 principle in access control management. For each dataset, multiple parameter configurations near the automated recommendations were evaluated, and the combination yielding the best balance between UPA coverage and role quality was selected. Final hyperparameter values and corresponding R_{var} are reported in Table IV.

Dataset	$\|U\|$	$n_{\textbf{comps}}$	R_{var}	ϵ	t_c
americas_large	3485	10	0.646	2	0.3
americas_small	3477	7	0.807	1	0.5
apj	2044	6	0.350	0.5	0.005
customer	10021	12	0.500	1	0.02
domino	79	4	0.840	1	0.1
emea	35	12	0.815	16	0.05
fire1	365	3	0.882	0.3	0.5
fire2	325	1	0.832	0.03	0.5
healthcare	46	3	0.841	1.5	0.4

Table IV: Determined hyperparameters

Table V compiles the results of the approach. The optimal number of roles used for comparison is provided by HP Labs [17] and reported by Blundo et al. [19]. Π represents regular permission expression, and π permission expression computed without outliers. The number of outliers is also reported.

Dataset	Π	π	no. out.	n_{roles}	n_{optimal}
americas_large	0.755	0.841	45	98	398
americas_small	0.881	0.913	31	21	178
apj	0.727	0.759	13	183	453
customer	0.798	0.812	58	47	276
domino	0.171	0.919	7	6	20
emea	0.615	0.799	4	25	34
fire1	0.905	0.974	14	4	64
fire2	0.994	1	7	1	10
healthcare	0.881	0.946	4	2	14

Table V: Results on real-world datasets

First, *americas_large*, *apj* and *customer* did not reach above 80% permission expression probably because of their sheer size and noise levels. Therefore, they required lower than average t_c to be expressed appropriately. The proposed approach yields over 75% permission expression on all datasets. Looking at the difference between Π and π, it is clear that on datasets like *domino* or *emea*, the small pool of identified outliers concentrate the majority of permission assignments, since π is above 80%, meaning clustered users have 80% of their permissions expressed. These users are likely administrators with an exceptionally high number of permission assignments, and therefore they were flagged as outliers. Overall, the number of identified outliers is between 1% and 10% of the number of users for all datasets, with a manageable absolute number that never exceeds 60 users. Also, the approach produced fewer roles across all datasets compared to the optimal number used for full expression, effectively dividing the number of roles by 4 on average. The exceptionally low amount of roles produced on *fire1* and *fire2* is likely due to a great number of similar users with a handful of exceptions.

IX. Discussion and Conclusion

This paper addresses the issue of role mining in the presence of privilege creep through three main contributions:

- A parameterizable synthetic data generator that aims to create realistic role mining datasets, with controlled noise and privilege creep injection.
- An effective cleaning methodology that removes noise and detects privilege creep in UPA matrices, achieving 90% detection accuracy and 95% legitimate permission retention on average.
- A comprehensive evaluation framework providing new metrics (LPR, PCC, PCDA, RCD) specifically designed for privilege creep-aware role mining.

A. Performance and practical impact

The proposed approach demonstrates strong performance on several datasets. On *americas_small*, *fire1*, *fire2*, and large synthetic datasets without user tension, the method expresses 90% of permission assignments while accurately removing noise and privilege creep. Notably, the approach reduces the number of required roles by up to a factor of ten, significantly improving access control manageability in large organizations. A key advantage of the proposed approach is its security-preserving nature [3], as no new permission assignments are added to the UPA, making it suitable for critical access review. The reunification phase improves role quality by removing potentially anomalous users out of the role mining phase.

Regarding privilege creep detection, Type I instances are detected with an accuracy above 90%, demonstrating the method's effectiveness for this common scenario. Type II privilege creep proves more challenging to detect as noise levels increase, with PCDA scores frequently falling below 50%. This occurs because higher noise levels render Type II privilege creep permission assignments statistically indistinguishable from random noise. However, due to the statistical analysis process used to clean the dataset, Type II privilege creep permissions are typically removed during the cleaning phase, even when not explicitly detected as privilege creep. While this maintains the cleaning objective, it highlights a limitation in the detection of project-based anomalies.

B. Limitations and areas for improvement

Hyperparameter sensitivity: The method's performance depends on manually tuning key parameters (k, ϵ, t_c). While heuristic estimates are provided for k and ϵ, no such guidance exists for t_c. This limitation is particularly evident on real-world datasets such as *apj, customer, domino,* and *emea*, which required unexpectedly low cleaning thresholds ($t_c < 0.3$) to achieve permission expression scores above 50%. This behavior stems from the interdependence of ϵ and t_c: targeting an outlier rate below 10% often necessitated inflated ϵ values, which caused cluster to merge and consequently

required lower-than-typical t_c thresholds to maintain adequate cleaning. These observations reveal a cascading effect amongst hyperparameters, with ϵ and especially t_c exhibiting the highest sensitivity. Visualizations of clusters with tools like t-distributed stochastic neighbor embedding (t-SNE) could guide hyperparameters selection.

Performance degradation on small datasets: Both performance evaluations reveal an inherent limitation: the approach demonstrates reduced effectiveness on datasets with fewer than 100 users approximately, such as *domino*, *emea*, *healthcare*, and synthetic datasets with user tension. On synthetic datasets, privilege creep detection accuracy averages only 60%, and legitimate permissions are erroneously removed. This degradation stems from the statistical properties of the clustering approach: DBSCAN requires sufficient sample density to reliably distinguish meaningful clusters from outliers, and small user populations provide insufficient data points, therefore flagging more users are outliers. The statistical threshold t_c, becomes unreliable when clusters contain too few members, as small anomalies within clusters are disproportionately represented.

Synthetic dataset generation: The data generator is based on a hierarchical tree structure with binomial distribution for users and permissions on nodes, which produces cleaner datasets than real-world ones, and exhibit artificial structural patterns that may not reflect the complexity and irregularities of actual enterprise environments. As such, the evaluation results on synthetic data are to be interpreted with caution.

Limited baseline comparison: Evaluation on real-world datasets uses HP Labs ideal results to evaluate role number reduction. However, it lacks comparison with state-of-the-art role mining techniques. Future work should primarily benchmark permission retention and privilege creep detection against methods in the related work section.

C. Future work

On the approach itself, user attributes could be used to enhance the clustering process by providing additional context. To address the hyperparameter sensitivity issue, adaptive thresholds based on cluster characteristics should be developed. Additionally, exploring alternative clustering algorithms, such as Isolation Forest or Local Outlier Factor (LOF), may improve performance on small datasets.

The synthetic data generator needs to be extended to model matrix-style and network-style organizational architectures, as well as separation of duties (SoD) constraints, enabling more realistic test scenarios. A comparative assessment against state-of-the-art role mining approaches would complete the validation of the method relative to existing techniques. Finally, mechanisms to assess and mitigate the operational impact of permission removal should be developed, to ensure the method's practical deployability in production environments.

ACKNOWLEDGMENT

We extend our gratitude to MITACS, Banque Nationale, Desjardins, Mondata, and Qohash for their invaluable support and contributions to this research.

REFERENCES

[1] J. Vaidya, V. Atluri, Q. Guo, and H. Lu, "Role mining in the presence of noise," in *IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, 2010, pp. 97–112.

[2] I. Molloy, N. Li, Y. Qi, J. Lobo, and L. Dickens, "Mining roles with noisy data," in *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*, 2010, pp. 45–54.

[3] J. Crampton, E. Eiben, G. Gutin, D. Karapetyan, and D. Majumdar, "Generalized noise role mining," in *Proceedings of the 27th ACM Symposium on Access Control Models and Technologies*, 2022, pp. 91–102.

[4] H. Kang, G. Liu, Q. Wang, Q. Zhang, J. Niu, and N. Luo, "An improved minimal noise role mining algorithm based on role interpretability," *Computers & Security*, vol. 127, p. 103 100, 2023.

[5] M. Frank, A. P. Streich, D. Basin, and J. M. Buhmann, "Multi-assignment clustering for boolean data," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 459–489, 2012.

[6] O. Durdag and A. Coskuncay, "Reconfiguring role-based access control via role clustering," *IEEE Access*, 2025.

[7] T. Wang and Q. Wu, "Role Minimization Optimization Algorithm Based on Concept Lattice Factor," *Mathematics*, vol. 11, no. 14, p. 3047, 2023.

[8] M. N. Nobi, R. Krishnan, Y. Huang, M. Shakarami, and R. Sandhu, "Toward Deep Learning Based Access Control," in *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '22, Association for Computing Machinery, 2022, pp. 143–154.

[9] S. Parkinson, S. Khan, J. Bray, and D. Shreef, "Creeper: A tool for detecting permission creep in file system access controls," *Cybersecurity*, vol. 2, no. 1, p. 14, 2019.

[10] S. Parkinson and S. Khana, "Identifying high-risk over-entitlement in access control policies using fuzzy logic," *Cybersecurity*, vol. 5, no. 1, p. 6, 2022.

[11] D. Alexander and D. K. Chikwarti, "Graph-Based AI Techniques for Role Mining and Access Optimization in Complex Enterprises," *International Journal of Advanced Engineering Technologies and Innovations*, vol. 1, no. 3, p. 519, 2023.

[12] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[13] D. Zhang, K. Ramamohanarao, and T. Ebringer, "Role engineering using graph optimisation," in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, 2007, pp. 139–144.

[14] J. Vaidya, V. Atluri, and J. Warner, "Roleminer: Mining roles using subset enumeration," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 144–153.

[15] H. Lu, J. Vaidya, and V. Atluri, "Optimal boolean matrix decomposition: Application to role engineering," in *2008 IEEE 24th international conference on data engineering*, IEEE, 2008, pp. 297–306.

[16] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo, "Evaluating role mining algorithms," in *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, 2009, pp. 95–104.

[17] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan, "Fast exact and heuristic methods for role minimization problems," in *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, 2008, pp. 1–10.

[18] F. Zhu, C. Yang, L. Zhu, and J. Gu, "Application of Matrix Factorization Role Mining Algorithm in Role-Based Access Control for Edge RMP," in *2024 9th International Conference on Electronic Technology and Information Science (ICETIS)*, 2024, pp. 761–767.

[19] C. Blundo and S. Cimato, "Role mining under user-distribution cardinality constraint," *Journal of information security and applications*, vol. 78, p. 103 611, 2023.

[20] Q. Guo and M. Tripunitara, "The Secrecy Resilience of Access Control Policies and Its Application to Role Mining," in *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*, Association for Computing Machinery, 2022, pp. 115–126.

[21] L. Dong, T. Wu, W. Jia, B. Jiang, and X. Li, "Computable Access Control: Embedding Access Control Rules Into Euclidean Space," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 10, pp. 6530–6541, 2023.