Découverte automatisée de hiérarchies de rôles pour les politiques de contrôle d'accès

Romuald THION
JEUNE CHERCHEUR

INSA-Lyon
Laboratoire LIRIS (UMR 5205)
Bât. Blaise Pascal (501)
20, Av. A. Einstein
69621 Villeurbanne CEDEX
romuald.thion@insa-lyon.fr

RÉSUMÉ. *L'identification des rôles dans un système - l'ingénierie des rôles - est une tâche essentielle dans l'établissement des politiques de contrôle d'accès basées sur les rôles (RBAC). La stratégie classique d'identification débute par une analyse fine du métier, à partir de laquelle sont dérivés les rôles. Cette approche, longue et coûteuse, impose une grande expertise du domaine et ne tire aucun profit des privilèges existants. Pour palier à ces défauts, la stratégie dite* role-mining *a été proposée. Basée sur l'agrégation des permissions existantes, cette approche automatise le processus d'ingénierie des rôles.*
Cet article présente une nouvelle technique de découverte automatisée des rôles et de leur relations hiérarchiques, à partir des privilèges existants dans le système. Notre proposition est fondée sur l'analyse de concepts formels dont le cadre théorique est bien caractérisé. Notre technique a été implémentée et expérimentée sur des jeux de données synthétiques et réels. Elle obtient de meilleurs résultats que les autres techniques proposées.

ABSTRACT. *Role-engineering is the task of discovering roles in a system. This task is essential in building efficient role-based access control policies. The classical approach in role-engineering is top-down. It relies on business process analysis to define roles from basic tasks, but ignore existing privileges. This approach is time consuming and expensive.*
This article presents a new automated role-engineering technique - a role-mining technique. Our bottom-up approach mines existing privileges to derive roles roles and their hierarchical relationships. Our efficient proposal relies on well-founded background and is money saving.

MOTS-CLÉS : *Ingénierie des rôles, contrôle d'accès à base de rôles, sous-hiérarchie de Galois*

KEYWORDS: *Role-engineering, Role-Based Access Control, Galois Sub-Hierarchy*

1. Introduction

Contrôler les accès, c'est déterminer si un *utilisateur* peut effectuer une *opération* demandée sur une *ressource*. Un droit d'opération sur une ressource et appelé *permission*, ou *privilège*. Une *politique de contrôle d'accès* définit les privilèges des utilisateurs d'un système. C'est un des principaux outils pour garantir la confidentialité d'un système, incontournable dans la mise en place d'une politique de sécurité.

Les politiques de Contrôle d'Accès (CA) basées sur les rôles, dites *RBAC* (Role Based Access Control) sont basées sur la constatation qu'une grande partie des décisions de CA sont déterminées par *l'autorité hiérarchique* ou *la fonction* du sujet dans son organisation [SAN 96] : cela forme le concept central de *rôle*. L'introduction de cette notion dans les politiques de CA comme intermédiaire entre les sujets et permissions facilite et simplifie la gestion de politiques concernant de nombreux utilisateurs.

La figure 1 (a) est la représentation graphique communément adoptée du modèle RBAC [FER 03]. Les acronymes URA et PRA signifient respectivement "User-Role Assignment" et "Permission-Role Assignment". Le principe est de ne pas affecter directement les privilèges aux utilisateurs, mais d'attribuer des rôles aux utilisateurs (URA), et des privilèges aux rôles (PRA). Le modèle RBAC est largement adopté et reconnu par les industriels et les académiques. Il est par exemple implémenté dans les principaux systèmes de de gestion de base de données du marché.

Le concept de hiérarchie de rôles (un rôle hérite des permissions de ses parents) diminue le nombre de permissions affectées aux rôles. Minimiser le nombre d'affectations à maintenir réduit les coûts d'administration des politiques et en améliore la fiabilité [FER 03]. La figure 1 (b) présente un exemple de hiérarchie composée de 5 rôles, où le moins privilégié est celui de *personnel hospitalier*. Dans cet exemple, les rôles *spécialiste* et *généraliste* héritent de *médecin*. Ainsi, tout utilisateur qui se voit attribué le rôle de *spécialiste*, dispose des privilèges affectés aux rôles *médecin* et *personnel hospitalier*.

L'ingénierie des rôles est la tâche qui consiste à identifier les concepts intermédiaires entres utilisateurs et permissions. Cette activité est entreprise lors de la mise en place de contrôle d'accès RBAC : elle est essentielle pour que l'établissement d'une politique RBAC soit rentable, mais c'est aussi l'activité la plus coûteuse et la plus longue. Un rapport du NIST [GAL 02] indique que le coût d'identification de chaque rôle se compte en milliers de dollars. Notre objectif est d'automatiser l'identification des rôles et de leur hiérarchie avec une technique qui soit complète, correcte et efficace, dans le but de *faciliter* et de réduire la *durée* et le *coût* de l'ingénierie des rôles.

Dans la section suivante, nous étudions les propositions existantes en ingénierie des rôles. Notre approche de découverte automatisée des rôles, de *role-mining* est basée sur le cadre formel de l'analyse de concepts formels, décrit en section 3. Notre proposition est détaillée en section 4. Les résultats expérimentaux seront présentés et comparés en section 5. Enfin, la dernière section conclue cet article et propose des perspectives.

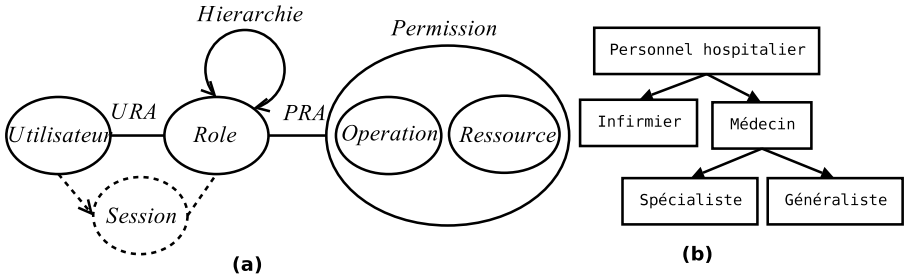

Figure 1. *Le modèle de référence RBAC (a) [FER 03], une hiérarchie de rôles (b)*

2. Travaux connexes

La thèse de Pete E. Epstein [EPS 02] est le papier fondateur de l'ingénierie des rôles, qui est définie comme "l'identification d'un ensemble de rôles qui soit *complet*, *correct* et *efficace*". Deux grandes stratégies d'identification des rôles sont étudiées :

– l'approche *descendante* est une méthodologie basée sur la *décomposition* des rôles en concepts intermédiaires. Cette approche se base sur la structure de l'organisation mais pas sur les privilèges existants. Elle a été appliquée avec succès dans de grandes structures [ROE 00], et approfondie avec la proposition basée sur des scenarii et de cas d'utilisation [NEU 02]. Cette approche, basée sur une modélisation rigoureuse des processus métiers n'a pas pas à notre connaissance été automatisée.

– l'approche *ascendante*, dite *role-mining* [KUH 03] est une identification des rôles *implicitement présents* dans le système, basée sur l'agrégation des permissions existantes, sans tenir compte de la structure de l'organisation. Elle a été proposée pour automatiser le processus d'ingénierie des rôles. La suite de cette section détaille les principales références [SCH 05, KUH 03, VAI 06] de cette approche.

La première proposition *SAM Role Miner* [KUH 03] est basée sur les outils IBM Intelligent Miner. La technique utilisée est celle du *demographic clustering* [GRA 02], hybride entre recherche de règles d'association et classification non-supervisée (clustering). Malheureusement, peu de détails sont donnés dans l'article et les algorithmes ne sont pas décrits, ce qui rend difficile la comparaison de cette approche avec d'autres. Les temps de calculs évoqués dans cette proposition sont de l'ordre de quelques heures pour environ 18.000 utilisateurs.

Plus récemment, l'algorithme de clustering hiérarchique *ORCA* [SCH 05] se fonde sur les forêts d'arbres comme représentation des hiérarchies de rôles. Cependant, au sens RBAC la hiérarchie est un *ordre partiel* sur les rôles, ce n'est pas nécessairement un arbre (dans le cas général, l'héritage multiple est autorisé : un rôle peut avoir plusieurs ancêtres), ce qui conduit à une limitation importante : une permission ne peut

Ensemble des objets :
$O = \{1, 2, 3, 4, 5, 6\}$

Ensemble des attributs :
$A = \{a, b, c, d, e, f, g, h\}$

	a rRes1	**b** wRes1	**c** xRes1	**d** rRes2	**e** wRes2	**f** xRes2	**g** rRes3	**h** wRes3
alice **1**		×	×	×	×			
bob **2**	×	×	×				×	×
charles **3**	×	×				×	×	×
david **4**				×	×			
eve **5**			×	×				
fred **6**	×							×

Tableau 1. *Exemple de contexte formel*

être associée qu'à des rôles d'un même arbre. La borne maximum du nombre de rôles obtenus n'est pas donnée. Un logiciel commercial *getRole*[1] a suivi ces travaux.

Dernièrement, une proposition a été faite par les auteurs de [VAI 06], c'est le travail le plus proche du notre. L'approche propose d'utiliser l'énumération de sous-ensembles de permissions pour identifier des rôles. Les auteurs proposent deux algorithmes : un complet, *CompleteMiner*, qui calcule tous les sous-ensembles de permissions possibles, et *FastMiner*, qui se limite au calcul d'intersections d'ensembles pris deux à deux. La borne maximum du nombre de rôles calculables n'est pas donnée et le critère utilisé pour sélectionner les rôles les plus pertinents est mal défini. Les auteurs proposent une méthodologie ayant pour but de vérifier la pertinence des rôles proposés par une mesure de précision (cf. section 5). Les jeux d'essais proposés pour l'évaluation de leur approche sont des cas favorables à *FastMiner*, l'algorithme *CompleteMiner* n'est pas évalué.

3. Analyse de concepts formels et sous-hiérarchie de Galois

Notre approche est basée sur des résultats issus des treillis de Galois et de l'analyse de concepts formels (dite *FCA* - Formal Concept Analysis). La notion de *concept formel* est une formalisation de la notion philosophique de concept [GAN 97]. La formalisation part d'un *contexte formel* : une relation binaire entre objets et attributs. Nous reprenons dans cette section les notations et les exemples de [ARÉ 07].

3.1. *Analyse de concepts formels*

Un *contexte formel* est un triplet $\mathbb{K} = (O, A, J)$ où O est l'ensemble des objets, A l'ensemble des attributs et $J \subseteq O \times A$ la relation d'incidence. Le tableau 1 est un exemple de contexte formel. Pour notre application de découverte de rôles, $\mathbb{K}$ est la

1. `http://www.getRole.de`

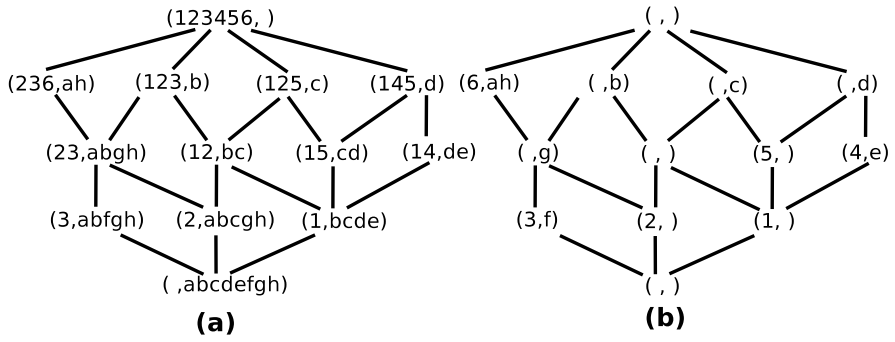

Figure 2. *Treillis complet (a) et réduit (b) associés au contexte formel du tableau 1*

matrice des droits d'accès du système, où les objets sont les utilisateurs (les lignes) et les attributs les permissions (les colonnes). La relation d'incidence indique quels sont les droits dont les utilisateurs disposent. Un exemple est présenté en tableau 1.

La relation d'incidence J induit deux applications α et ω qui relient 2^O (l'ensemble des sous-ensembles d'objets) et 2^A (l'ensemble des sous-ensembles d'attributs). α (resp. ω) s'interprète comme l'application qui à un ensemble d'objets (resp. d'attributs) fait correspondre l'ensemble des attributs (resp. des objets) avec lesquels *tous* les objets (resp. attributs) de l'ensemble sont en relation. Par exemple, d'après le tableau 1, $\alpha(23) = abgh$[2] signifie que les utilisateurs *bob* et *charles* disposent tous deux des droits de lecture et d'écriture sur la ressource 1 (*rRes1*, *wRes1*) ainsi que des droits de lecture et d'écriture sur la ressource 3 (*rRes3*, *wRes3*).

$$\forall X \in 2^O, \alpha(X) = \{y \in A \mid \forall\, x \in X, (x, y) \in J\}$$
$$\forall Y \in 2^A, \omega(Y) = \{x \in O \mid \forall\, y \in Y, (x, y) \in J\}$$

Une paire d'ensembles en correspondance $c = (X, Y)$ tels que $Y = \alpha(X)$ (ou $X = \omega(Y)$) est appelée *concept formel*. La FCA définit les concepts comme des ensembles *maximaux* d'objets partageant les mêmes attributs. L'ensemble X des objets d'un concept est appelé *extension* (complète), l'ensemble Y des attributs est appelé *intention* (complète). Par exemple $(12, bc)$ est un concept dont l'extension est 12 et son intention bc, par contre, $(2, bc)$ n'est pas un concept car $2 \neq \omega(bc)$.

L'ensemble des concepts C muni de la relation d'ordre partiel $\leq_C$: $(X_1, Y_1) \leq_C (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2\ (ou\ Y_2 \subseteq Y_1)$ forme un treillis de Galois appelé *treillis de concepts* (cf. figure 2). Le théorème fondamental de la FCA est que l'ensemble de tous les concepts qui peuvent être construits sur un contexte formel donné forme un treillis complet quand il est ordonné par l'inclusion ensembliste.

2. on note l'ensemble $\{a_1, \ldots, a_n\}$ par $a_1 \ldots a_n$ pour simplifier la notation

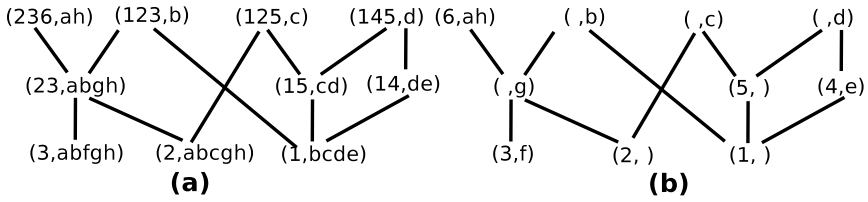

Figure 3. *Diagrammes de Hasse de la SHG complète (a) et réduite (b)*

Pour l'application que nous proposons, un concept formel est assimilé à un rôle, reliant un ensemble d'utilisateurs (son extension) à un ensemble de permissions (son intention). La relation d'ordre $\leq_C$ est assimilée à l'ordre partiel entre les rôles munis de la relation d'héritage. Ainsi $(12, bc)$ est un rôle dont dispose les utilisateurs *alice* et *bob*, ce rôle conférant les droits d'écriture et d'exécution sur la ressource 1.

Un concept C' est dit ancêtre (resp. descendant) d'un concept C ssi $C' <_C C$ (resp $C' >_C C$). Un objet (resp. un attribut) x est dit être *introduit* par un concept si x fait partie de l'extension (resp. de l'intention) de ce concept et qu'aucun ancêtre (resp. descendant) de ce concept ne contient x dans son extension (resp. intention). C'est le concept-objet (resp. concept-attribut) de x. Cette notion nous est utile pour différencier les permissions accordées *explicitement* aux rôles (elle sont introduites) de celles transmises par la relation d'héritage. Par exemple, le concept $(23, abgh)$ introduit l'attribut g, le droit de lecture sur la ressource 3 est explicitement attribué au rôle $(23, abgh)$.

Chacun des 13 nœuds des treillis de la figure 2 sont les concepts formels construits à partir du contexte du tableau 1. La sous-figure 2 (a) est le treillis *complet*, c'est à dire qui comporte l'extension et l'intention complète de chaque concept. La sous-figure 2 (b) est le treillis *réduit*, qui ne comporte que les extensions et intentions réduites, c'est à dire les nouveaux objets/attributs *introduits* par les concepts.

3.2. *Sous-hiérachie de Galois*

La *sous-hiérachie de Galois* (SHG) a été introduite par Godin [GOD 95] dans le cadre de la restructuration de hiérarchies de classes, pour réduire le nombre des concepts. La SHG est un sous ordre particulier de $(C, \leq_C)$ qui ne *contient que* les concepts-objets et les concepts-attributs du treillis de Galois. Les objets introduits par un concept forment son *extension réduite*, ses attributs son *intention réduite*. La SHG supprime donc les concepts qui ne sont *ni concepts objets ni concepts attributs* : les concepts $\{\emptyset, \emptyset\}$ de la figure 2 (b). La figure 3 présente les diagrammes de Hasse (une représentation visuelle d'un ordre fini qui en facilite la compréhension) de la SHG complète (a) et réduite (b).

Notion FCA	***Notion RBAC***	***Symb.***
concept	rôle	C
objet	utilisateur	O
attribut	permission	A
relation d'ordre partiel	relation d'héritage	$\leq_C$
extension réduite	utilisateurs *explicitement* affectés à un rôle	e
intention réduite	permissions *explicitement* affectées à un rôle	i
extension complète	utilisateurs *implicitement* affectés à un rôle : affectés à ses descendants	E
intention complète	permissions *implicitement* affectées à un rôle : affectées à ses ancêtres	I
concepts ancêtres	rôles dont hérite un rôle	a
concepts descendants	rôles héritant d'un rôle	d

Tableau 2. *Identification des termes de FCA et de RBAC*

Les auteurs de [ARÉ 07] ont proposé un algorithme *Pluton* de calcul de la SHG d'un contexte formel. Cet algorithme fait suite à [BER 05] qui propose une technique issue des graphes pour calculer une extension linéaire de la sous-hiérarchie de Galois. *Pluton* est décomposé en 3 sous-algorithmes *TomThumb*, *ToLinext* et *ToGSH*. *TomThumb* prend en entrée un contexte formel et retourne une liste ordonnée des labels présents dans la SHG (cf. figure 3). Cette extension linéaire est un ordre total compatible avec l'ordre partiel $(C, \leq_C)$. *ToLinext* fusionne les labels pour former les concepts, et enfin l'algorithme *ToGSH* calcule la SHG. La complexité en temps théorique de *TomThumb* est évaluée en $\mathcal{O}(|J|)$, l'implémentation brutale de *ToLinext* est décrite dans [BER 05] avec une compléxité de $\mathcal{O}((|O| + |A|)^3)$, enfin *ToGSH* a été évalué en in $\mathcal{O}((|O| + |A|)^2 * max(|O|, |A|)^2)$.

3.3. *Le choix de l'analyse de concepts formels*

La recherche des ensembles fermés, dont font partie les concepts formels, est un domaine de recherche actif, en France notamment [PAS 99, NOU 02]. Cependant, un des principaux freins à l'utilisation de l'ensemble des concepts formels d'une relation binaire, est que leur nombre peut être exponentiel au pire des cas, la borne maximum étant $2^{min(|O|,|A|)}$ [GAN 97]. Ces algorithmes sont donc coûteux et peuvent s'avérer difficilement utilisables sur de grands jeux de données. De plus, parmi l'ensemble des concepts formels, certains n'apportent aucune information supplémentaire (ils n'introduisent ni attributs ni objets) et rendent difficile l'interprétation des résultats. C'est pour résoudre ces problèmes qu'a été proposée la SHG. Pour l'application à la découverte de rôles proposée, la SHG est un cadre adéquat car :

– la notion de *concept* telle que définie dans la FCA est semblable à la notion de rôle telle que proposée dans [SAN 96, FER 03] : une abstraction qui peut être définie soit par les individus qui lui sont attachés, soit par les propriétés dont elle dispose.

– les concepts sont des *regroupements maximaux* d'objets et d'attributs, ce qui répond à un besoin en terme d'ingénierie : les rôles doivent comporter le maximum d'utilisateurs et de permissions,

– la SHG bénéficie d'études théoriques et d'une *caractérisation formelle*,

– l'algorithme *Pluton* est *performant* : les résultats expérimentaux montrent que l'on peut calculer la SHG en temps interactif (moins de 3 secondes) pour quelques milliers d'utilisateurs et des centaines de permissions,

– la SHG comporte *toutes* les intersections/unions de concepts apportant de l'information.

4. Découverte automatisée de hiérarchies de rôles

L'approche que nous proposons est basée sur l'identification des notions *FCA* présentées en section 3 (objets, attributs, concepts, sous-hiérarchie de Galois) aux notions considérées dans les politiques de contrôle d'accès *RBAC* (utilisateurs, permissions, rôles, hiérarchie de rôles). Ainsi, l'ensemble O des objets est l'ensemble des utilisateur du système, les attributs sont les permissions et les concepts calculés sont les rôles organisés en hiérarchie, c'est à dire les regroupements maximaux d'utilisateurs qui partagent les même permissions. Le tableau 2 synthétise ces identifications entre vocabulaire des *concepts formels* et des *politiques RBAC*. La colonne ***Symb.*** de ce tableau est le symbole associé que nous utiliserons dans la suite de l'article.

4.1. *Description de l'approche*

Nous désignons par *l'expert*, la où les personnes responsable(s) de l'ingénierie de rôles dans un système, en vue de la mise en place d'une politique de contrôle d'accès RBAC. Notre approche est décomposée en 5 étapes :

1) obtenir la *matrice des droits* d'accès du système, tableau 1,

2) calculer la *sous-hiérarchie de Galois* associée à l'aide de *Pluton*, qui comporte les rôles potentiels du système[3], figure 3,

3) *classer les concepts* obtenus selon une mesure de pertinence choisie, tableau 3,

4) *élaguer* la SHG selon le classement obtenu afin de réduire sa taille en vue d'obtenir un diagramme de Hasse représentable, figure 4,

5) proposer cette hiérarchie de concepts à l'expert pour évaluation et analyse.

3. Notons que dans le cas d'une hiérarchie de rôles *parfaite*, c'est à dire où un utilisateur est affecté à un seul rôle et où au plus un rôle dispose de chaque permission, la SHG calculée est exactement la hiérarchie utilisée et qu'on ne peut supprimer aucun concept.

	C	$\shortmid e \shortmid$	$\shortmid \mathbf{i} \shortmid$	$\shortmid e \shortmid \times \shortmid i \shortmid$	$\mid E \mid$	$\mid I \mid$	$\mid E \mid \times \mid I \mid$	$\shortmid a \shortmid$	$\shortmid d \shortmid$
	$\{6, ah\}$	*1*	**2**	2	3	2	6	0	1
	$\{3, f\}$	*1*	**1**	1	1	5	5	1	0
	$\{4, e\}$	*1*	**1**	1	2	2	4	1	1
	$\{\emptyset, g\}$	*0*	**1**	0	2	4	8	2	2
	$\{\emptyset, b\}$	*0*	**1**	0	3	1	3	0	2
	$\{\emptyset, c\}$	*0*	**1**	0	3	1	3	0	2
	$\{\emptyset, d\}$	*0*	**1**	0	3	1	3	0	2
	$\{2, \emptyset\}$	*1*	**0**	0	1	5	5	2	0
	$\{1, \emptyset\}$	*1*	**0**	0	1	4	4	3	0
	$\{5, \emptyset\}$	*1*	**0**	0	2	2	4	2	1
Σ	*10*	***6***	***8***	*4*	*21*	*27*	*45*	*11*	*11*

Tableau 3. *Classement des concepts de la SHG de la figure 3 selon* $\shortmid$ **i** $\shortmid$ *puis* $\shortmid e \shortmid$

4.2. *Classement des concepts*

Le nombre de concepts de la SHG peut être supérieur au nombre d'utilisateurs du système, il faut donc réduire ce nombre en supprimant les concepts les moins pertinents comme rôles potentiels. Le choix d'une mesure de pertinence est un problème délicat et difficile à résoudre dans le cas général. Selon les organisations, l'expert peut privilégier de minimiser le *nombre de rôles* dans la hiérarchie, le nombre *d'affectations de permissions* aux rôles ou encore le nombre *d'affectations de rôles* aux utilisateurs. Le tableau 2 présente les critères implémentés dans notre prototype pour bâtir une mesure de pertinence. Le tableau 3 présente un exemple de classement des concepts de la SHG de la figure 3 selon le nombre de permissions affectées au rôle $\shortmid$ **i** $\shortmid$ puis par nombre d'utilisateurs affectés $\shortmid e \shortmid$. La somme des $\shortmid$ **i** $\shortmid$ est égale à $\mid O \mid$, celle des $\shortmid e \shortmid$ est égale à $\mid A \mid$.

4.3. *Elaguage de la sous-hiérarchie de Galois*

Une fois les concepts ordonnés, nous supprimons de la SHG les concepts considérés comme les moins pertinents. Comme nous voulons que notre approche soit correcte, il faut que la SHG élaguée comporte *toute* l'information contenue dans le contexte formel d'origine : il ne faut *ni supprimer ni ajouter* des droits aux utilisateurs du système. Pour garantir cette propriété, nous procédons ainsi :

1) calculer une matrice outil M de taille $|O| \times |A|$, qui stocke pour chaque couple $objet \times attribut$ le nombre de concepts où il est présent. Par exemple, $M[1, c] = 3$ car l'objet 1 et l'attribut c apparaîssent trois fois ensemble (dans les concepts $(15, cd)$, $(125, c)$ et $(1, bcde)$),

2) pour chaque concept $c = (O_c, A_c)$, du moins pertinent au plus pertinent, si $\forall o \in O_c, \forall a \in A_c, M[o,a] > 1$, alors supprimer c car il n'entraîne pas de perte d'information et mettre à jour M : $\forall o \in O_c, \forall a \in A_c, M[o,a] \leftarrow M[o,a] - 1$

3) une fois la liste des concepts réduite, utiliser l'algorithme *ToGSH* [ARÉ 07] pour reconstruire le graphe simplifié $G' = (V', E')$. *ToGSH* est un des trois sous-algorithmes qui composent *Pluton* : il prend en entrée une liste de concepts et un contexte formel pour construire G'.

L'algorithme d'élagage supprime les 3 plus petits concepts du tableau 3, obtenant la hiérarchie de rôles de la figure 4. Chaque rôle est représenté par un losange, avec comme label les d'utilisateurs qui l'endossent et les permissions qui lui sont affectées. Les flèches indiquent les relations d'héritages entre rôles, du moins privilégié au plus. Associée à chaque rôle, une note indique le nombre d'utilisateurs qui l'endossent explicitement et implicitement (entre parenthèse) et le nombre de permissions affectées. Cette hiérarchie est composée de *7* rôles et de *4* relations d'héritage. Seuls les utilisateurs *1* et *2* endossent plus d'un rôle.

5. Résultats

Nous avons intégré l'algorithme *Pluton*, les générateurs de politiques et l'élagage de la SHG dans un prototype écrit en C++, utilisant la Standard Template Library (STL) ainsi que la bibliothèque Boost Graph Library[4]. La suite Graphviz[5] a été utilisée pour la représentation graphique des diagrammes de Hasse. La machine qui a servi aux expérimentations est un Pentium 4 3GHz, 1 Go, KUbuntu 6.10 (2.6.17), gcc 4.1.2.

5.1. *Jeux d'essai synthétiques*

[VAI 06] propose une méthodologie pour mesurer la pertinence et l'efficacité des approches ascendantes d'ingénierie de rôles. Cette approche, basée sur les principes du reverse-engineering, consiste à générer une politique RBAC, calculer la matrice des droits d'accès à partir de cette politique puis vérifier si les rôles calculés à partir du contexte formel sont bien ceux initialement générés :

– générer aléatoirement une politique RBAC de n rôles sans hiérarchie (nous avons complété la méthodologie pour qu'elle prenne en compte la hiérarchie des rôles),

– construire la matrice des droits d'accès à partir de cette politique,

– exécuter l'algorithme de découverte des rôles sur la matrice des droits d'accès,

– classer les concepts calculés selon une mesure de pertinence,

– parmi les n premiers concepts du classement, compter ceux qui sont des rôles générés : la mesure de précision en % .

4. `www.boost.org/libs`
5. `www.graphviz.org`

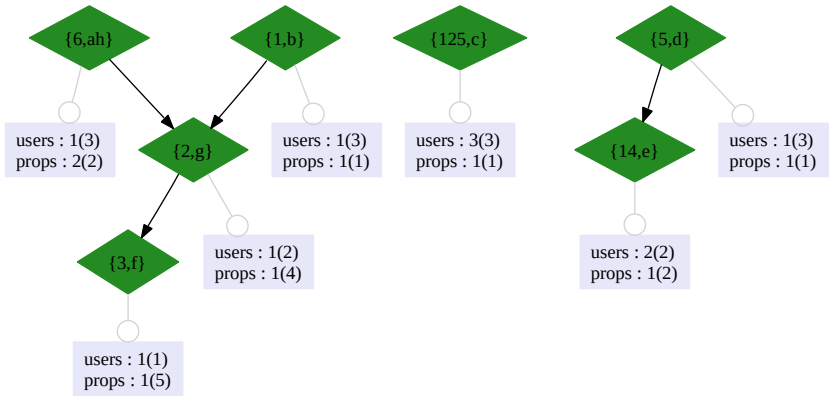

Figure 4. *Hiérachie de rôles obtenue à partir du contexte formel du tableau 1*

La figure 5 présente les résultats obtenus par notre proposition. Les expérimentations sont décrites dans le tableau 4. Le générateur de politique a comme paramètres le nombre de rôles `NR`, d'utilisateurs `NU`, de permissions `NP`, le nombre maximum de rôles `MRU` par utilisateurs ainsi que le nombre maximum de permissions par rôle `MPR`. Pour chaque jeu d'essai, 4 courbes sont proposées :

– P (f) : la précision en % (sans hiérarchie),

– T (f) : le temps d'exécution en secondes (sans hiérarchie),

– P (h) : la précision en % (avec hiérarchie),

– T (h) : le temps d'exécution en secondes (avec hiérarchie),

Les jeux **a**, **b** et **c** de la figure 5. La mesure de pertinence utilisée pour le classement des concepts est la taille de l'extension réduite (e du tableau 2), elle se rapproche de la mesure choisie pour *FastMiner* appelée *prioritization*. Nous obtenons des mesures de précision légèrement supérieures à *FastMiner* pour le jeu **a**, supérieures pour le jeu **b** et nettement supérieures pour le jeu **c**. Pour le jeu d'essai **d**, la précision décroît avec l'augmentation du nombre de rôles attribués à chaque utilisateur. Les rôles générés ne sont pas les premiers du classement des concepts, mais sont cependant majoritairement présents dans la SHG : la mesure de pertinence par taille de l'extension réduite n'est donc pas adaptée à ce jeu d'essai.

Notre approche est entre *50 et 100 fois* plus rapide que les performances de *FastMiner* présentées dans [VAI 06], nous ne disposons pas des résultats de *CompleteMiner*. Même si le choix du langage influe sur les performances (C++ pour nous, Java pour *FastMiner*), la différence est suffisamment significative pour qu'elle ne soit pas uniquement attribuée à l'implémentation. Les résultats avec un générateur de politique avec ou sans hiérarchie sont assez similaires, le calcul est sensiblement plus coûteux dans le cas des hiérarchies car le contexte formel est plus dense.

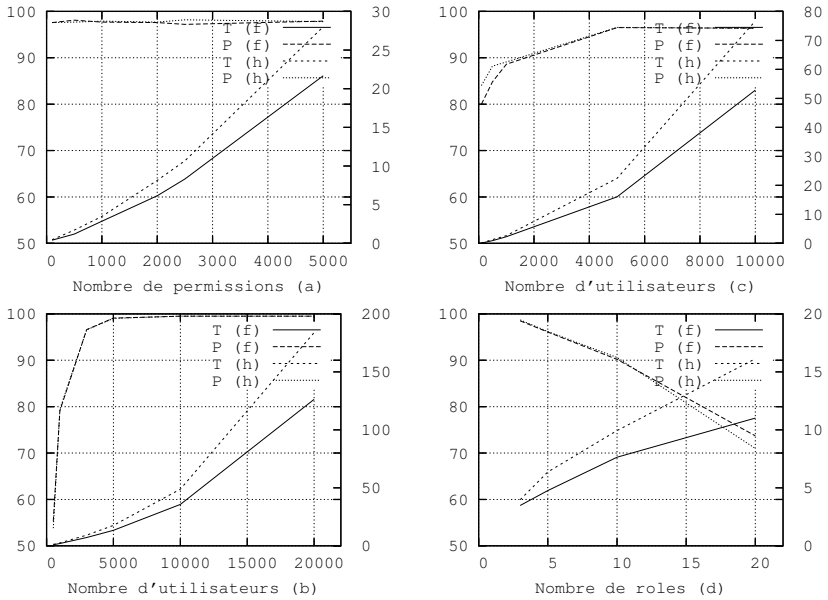

Figure 5. *Résultats expérimentaux, moyenne sur 10 exécutions*

Nous avons également évalué les performances sur les jeux d'essai proposés par la FIMI (Frequent Itemset Mining Implementations), qui met à disposition plusieurs jeux d'essai volumineux pour comparer les algorithmes de data-mining. Par exemple, pour le jeu "connect"[6] de taille 67557 objets et 129 attributs, la SHG est calculée en un peu moins de 30 minutes.

5.2. *Jeux d'essai réels*

Nous avons eu à notre disposition deux matrices des droits issues d'une même organisation (un établissement hospitalier). Le premier jeu d'essai est la matrice d'accès aux applications de l'établissement, qui indique parmi les *12* applications existantes, quelles sont celles autorisées pour chacun des *1582* utilisateurs de l'organisation. La SHG calculée à partir de ce jeu d'essai comporte *48 concepts et 93 relations* hiérarchiques : une politique difficile à représenter graphiquement et inutilisable en pratique. L'élagage maximum conduit à une politique de *12 rôles sans hiérarchie* : il s'agit du cas limite où chaque rôle donne exactement une permission. Le temps de calcul pour ce jeu d'essai est de *0.05 seconde* environ.

6. `http://fimi.cs.helsinki.fi/data/connect.dat`

Jeu	*Description*	*Paramètres*
a	Nombre d'utilisateurs par rôle constant, nombre de permissions variable	`MRU=3, NR=100, NU=2000`
b	Nombre de permissions par rôle constant, nombre de d'utilisateurs variable	`MRU=3, MPR=150, NR=200, NP=1500`
c	Nombre de permissions constant, nombre d'utilisateurs par rôle variable	`MRU=3, MPR=150, NP=1500`
d	Nombre d'utilisateurs, de permissions et de rôles constants, nombre de rôles par utilisateur variable	`NU=2500, NR=100, MPR=50, NP=1500`

Tableau 4. *Paramètres utilisés pour les expérimentations*

Le second jeu d'essai est une matrice des droits d'accès d'une des applications de l'établissement. Ce jeu d'essai ne concerne ainsi que *37* utilisateurs et les *184* permissions spécifiques à cette application. La SHG calculée à partir de ce jeu d'essai comporte *33 concepts et 117 relations* hiérarchiques. Le cas limite pour cet exemple est une politique de *8* rôles avec *2* relations d'héritage. Le temps de calcul pour ce jeu d'essai est de *0.01 seconde* environ.

Le choix d'une mesure de pertinence et du nombre de rôles minimum que l'on souhaite permet de calculer des politiques de taille variables, comprises entre les deux extrêmes que sont les cas limites et la SGH complète. Ces jeux d'essai sont intéressants car les proportions des contextes sont très différentes et comportent des cas limites. Par exemple, parmi les *12* applications du premier jeu, deux d'entre elles ne sont utilisées que par exactement une seule personne chacune. Cette attribution de droit crée ainsi deux concepts spécifiques à ces utilisateurs, qui *ne peuvent pas être supprimés* par l'algorithme d'élagage.

5.3. *Interprétation des résultats obtenus*

Au regard des résultats obtenus, trois grandes catégories de rôles apparaissent :

– les rôles *pertinents* : il s'agit des concepts à la fois concepts-attributs et concepts-objets. Ces rôles sont les plus significatifs : ils apportent explicitement des permissions et des utilisateurs les endossent également explicitement.

– les rôles *abstraits* : il s'agit des concepts-attributs : des rôles auxquels *aucun utilisateur n'est affecté directement*, les utilisateurs étant affectés soit à leur ancêtres, soit à leurs descendants dans la hiérarchie. Une grande partie de ces rôles est à supprimer, mais il est intéressant d'en garder certains pour leur *pouvoir structurant* : ils peuvent faciliter la compréhension de la hiérarchie,

– les rôles *spécifiques* à un groupe d'utilisateurs : il s'agit des concepts-objets, ces rôles sont à éviter. Il vaut mieux privilégier l'affectation multiple (un utilisateur

peut disposer de plusieurs rôles) plutôt que d'avoir à maintenir *un rôle spécifique* à un groupe d'utilisateur qui n'apporte *aucune permission supplémentaire* que celles de ses ancêtres.

6. Discussion et conclusion

Cet article présente une technique ascendante qui automatise le processus d'ingénierie des rôles. Basé sur la notion de sous-hiérarchie de Galois, notre proposition calcule, selon un critère paramétrable, une hiérarchie de rôles correcte (qui ne modifie pas les permissions attribués aux utilisateurs) à partir des privilèges existants d'un système. La technique employée est plus rapide et plus précise que celles proposées par les auteurs de [VAI 06].

6.1. *Obtention de la matrice des droits*

Notre proposition identifie les rôles implicitement présents dans la matrice des droits d'accès d'un système, or, pour des raisons pratiques évidentes, les droits des utilisateurs ne sont pas directement gérés avec une telle matrice : il faut dériver et croiser des informations existantes sur les privilèges pour l'obtenir. Le cas des systèmes utilisant des politiques à base de liste, comme les Access Control Lists (ACL) ou Capability Lists (CL), est facile à traiter. Chaque liste représente soit une colonne (ACL) ou une ligne (CL) de la matrice des droits. Cependant, il peut exister des *dépendances implicites* entre les permissions du système. Par exemple, il existe des relations implicites entre les privilèges dans le SGBD Oracle dues à la différence entre privilèges *systèmes* et *objets* : donner le droit de création de table à un utilisateur lui attribue implicitement les autres droits de manipulation de données sur cette table.

6.2. *Mesure de compléxité d'une politique*

La mesure de la complexité des politiques RBAC a été abordée dans [JAE 01], elle pourrait avantageusement remplacer la mesure de pertinence utilisée pour comparer la qualité des propositions d'ingénierie des rôles. Malheureusement, comme pour la pertinence d'un rôle, le problème est délicat et fortement dépendant de l'organisation qui déploie une politique RBAC. Alors qu'une organisation considèrera qu'il vaut mieux minimiser le nombre de relations hiérarchiques pour simplifier une politique, une autre préférera minimiser le nombre de rôles dont un utilisateur dispose.

6.3. *Aspects théoriques*

Parmi les concepts présent dans la SHG, certains sont *sup-irréductibles* ou *inf-irréductibles* [GAN 97] : ils ne peuvent pas être exprimés comme l'union ou l'inter-

section d'autres concepts. Nous pensons qu'il serait intéressant d'introduire cette notion dans notre proposition : à partir d'une caractérisation formelle nous apposerions des *labels* sur les concepts de la SHG afin de faciliter la sélection des concepts pertinents et l'interprétation de la hiérarchie de rôles obtenue. De plus, la notion d'élagage que nous utilisons pourrait être intégrée à un algorithme comme *Pluton*, pour aboutir à la définition d'un algorithme de construction de la SHG sous contraintes, comme il en existe pour la construction du treillis des concepts [BES 04].

6.4. *Vers un atelier complet d'ingénierie des rôles*

Nous pensons qu'il est possible de proposer un atelier complet d'ingénierie des rôles basé sur la FCA, comme il en existe déjà pour le génie logiciel [TIL 03]. Les graphes conceptuels, étroitement liés à la FCA [WIL 97], sont candidats à une représentation adéquate des politiques RBAC [THI 06b] et pourraient être intégré à un tel atelier. Enfin, l'intégration d'autres propositions attenantes à RBAC, comme la vérification des politiques [THI 06a] permettrait de proposer un outil avec lequel superviser la gestion des politiques RBAC de bout en bout : de l'ingénierie des rôles jusqu'à l'administration quotidienne en passant par la vérification.

Remerciements

L'auteur remercie cordialement André, Nadine et leurs collaborateurs pour les jeux de données et leur accueil amical.

7. Bibliographie

[ARÉ 07] ARÉVALO G., BERRY A., HUCHARD M., PERROT G., SIGAYRET A., « Comparison of Performances of Galois Subhierarchy-building Algorithms », *5th International Conference on Formal Concept Analysis, ICFCA'07*, vol. 4390 de *LNCS*, 2007.

[BER 05] BERRY A., HUCHARD M., MCCONNELL R. M., SIGAYRET A., SPINRAD J., « Efficiently Computing a Linear Extension of the Sub-hierarchy of a Concept Lattice. », *3rd International Conference on Formal Concept Analysis, ICFCA'2005*, vol. 3403 de *LNCS*, 2005, p. 208-222.

[BES 04] BESSON J., ROBARDET C., BOULICAUT J.-F., « Constraint-Based Mining of Formal Concepts in Transactional Data », *Advances in Knowledge Discovery and Data Mining, PAKDD'2004*, vol. 3056 de *LNCS*, 2004, p. 615-624.

[EPS 02] EPSTEIN P. E., « Engineering of Role/Permissions assignments », PhD thesis, Fairfax University, VA, USA, 2002.

[FER 03] FERRAIOLO D., KUHN R., CHANDRAMOULI R., *Role-Based Access Control*, Artech House Publishers, 2003.

[GAL 02] GALLAHER M. P., O'CONNOR A. C., KROPP B., « The Economic Impact of Role-Based Access Control », rapport, 2002, Planning report 02-1, National Institute of Standards

and Technology.

[GAN 97] GANTER B., WILLE R., *Formal Concept Analysis : Mathematical Foundations*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997, Translator-C. Franzke.

[GOD 95] GODIN R., MINEAU G., MISSAOUI R., MILI H., « Méthodes de Classification Conceptuelle Basées sur les Treillis de Galois et Applications », *Revue d'Intelligence Artificielle*, vol. 9, n° 2, 1995, p. 105-137.

[GRA 02] GRABMEIER J., RUDOLPH A., « Techniques of Cluster Algorithms in Data Mining », *Data Min. Knowl. Discov.*, vol. 6, n° 4, 2002, p. 303–360, Kluwer.

[JAE 01] JAEGER T., « Managing access control complexity using metrices », *6th ACM symposium on Access control models and technologies, SACMAT'01*, New York, NY, USA, 2001, ACM Press, p. 131–139.

[KUH 03] KUHLMANN M., SHOHAT D., SCHIMPF G., « Role mining - revealing business roles for security administration using data mining technology », *8th ACM symposium on Access control models and technologies, SACMAT'03*, ACM Press, 2003, p. 179–186.

[NEU 02] NEUMANN G., STREMBECK M., « A scenario-driven role engineering process for functional RBAC roles », *7th ACM symposium on Access control models and technologies, SACMAT'02*, New York, NY, USA, 2002, ACM Press, p. 33–42.

[NOU 02] NOURINE L., RAYNAUD O., « A fast incremental algorithm for building lattices. », *J. Exp. Theor. Artif. Intell.*, vol. 14, n° 2-3, 2002, p. 217-227.

[PAS 99] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Discovering Frequent Closed Itemsets for Association Rules. », BEERI C., BUNEMAN P., Eds., *ICDT*, vol. 1540 de *LNCS*, Springer, 1999, p. 398-416.

[ROE 00] ROECKLE H., SCHIMPF G., WEIDINGER R., « Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. », *5th ACM Workshop on Role-Based Access Control,RBAC'00*, 2000, p. 103-110.

[SAN 96] SANDHU R. S., COYNE E. J., FEINSTEIN H. L., YOUMAN C. E., « Role-Based Access Control Models. », *IEEE Computer*, vol. 29, n° 2, 1996, p. 38-47.

[SCH 05] SCHLEGELMILCH J., STEFFENS U., « Role mining with ORCA », *10th ACM symposium on Access control models and technologies, SACMAT'05*, New York, NY, USA, 2005, ACM Press, p. 168–176.

[THI 06a] THION R., COULONDRE S., « Modeling and Inferring on Role-Based Access Control Policies Using Data Dependencies. », *17th International Conference on Database and Expert Systems Applications, DEXA'06*, LNCS, 2006, p. 914-923.

[THI 06b] THION R., COULONDRE S., « Representation and Reasoning on Role-Based Access Control Policies with Conceptual Graphs. », *14th International Conference on Conceptual Structures, ICCS'06*, vol. 4068 de *LNCS*, 2006, p. 427-440.

[TIL 03] TILLEY T., COLE R., BECKER P., EKLUND P., « A Survey of Formal Concept Analysis Support for Software Engineering Activities », STUMME G., Ed., *1st International Conference on Formal Concept Analysis, ICFCA'03*, Springer-Verlag, February 2003.

[VAI 06] VAIDYA J., ATLURI V., WARNER J., « RoleMiner : mining roles using subset enumeration », *13th ACM conference on Computer and Communications Security, CCS'06*, New York, NY, USA, 2006, ACM Press, p. 144–153.

[WIL 97] WILLE R., « Conceptual Graphs and Formal Concept Analysis », *5th International Conference on Conceptual Structures, ICCS'97*, 1997, p. 290-303.