UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Interface graphique pour la représentation de contrôle de flux de données sécuritaires

MÉMOIRE PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DU PROGRAMME DE MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

PAR

Mohamed Lamine Ripama

Juillet 2025

Jury d'évaluation

Dr. Kamel Adi…… Président du Jury

Dr. Omer Nguena Timo……Membre du Jury

Dr. Luigi Logrippo…… Directeur de recherche

TABLE DES MATIÈRES

TABLE DES MATIÈRES i

LISTES DES TABLEAUX vi

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES vii

RÉSUMÉ ix

ABSTRACT x

CHAPITRE 1 1

1.1. INTRODUCTION 1

1.2. CONTEXTE ET PROBLÉMATIQUE 2

1.3. L'IMPORTANCE DE LA VISUALISATION DES REALATIONS ENTITÉS-PERMISSIONS 4

1.4. OBJECTIF PRINCIPALE 4

1.5. OBJECTIF SPÉCIFIQUE 5

1.6. CONTRIBUTION VISÉE 5

1.7. ORGANISATION DU MÉMOIRE 6

CHAPITRE 2 : REVUE DE LA LITÉRATURE 8

2.1. INTRODUCTION 8

2.2. MODÈLE D'ACCÈS ET SÉCURITÉ 8

2.2.1. Modèle Discrétionnaire d'Accès (DAC) 10

2.2.1.1. Modèle de Lampson 10

2.2.1.2. Conclusion dur les modèles discrétionnaires 12

2.2.2. Modèle d'accès multi-niveaux (MAC) 13

2.2.2.1. Modèle de Bell-LaPadula 13

2.2.2.2. Modèle d'intégrité de Biba 16

2.2.2.3. Modèle de Brewer et Nash 19

2.2.2.4. Conclusion sur le modèle d'accès multi-niveaux 21

2.2.3. Modèle RBAC 21

2.2.3.1. Principes du modèle RBAC 22
2.2.3.2. Noyau RBAC (RBAC 0) 23
2.2.3.3. Conclusion sur le modèle RBAC 24
2.2.4. Modèle ABAC 25
2.2.4.1. Conclusion sur le modèle ABAC 28
2.3. CONCLUSION 29
CHAPITRE 3 : CADRE THÉORIQUE 30
3.1. INTRODUCTION 30
3.2. CADRE THEORIQUE GÉNÉRAL 31
3.2.1. Définition des concepts de base 31
3.2.2. Calcul de relations et structures 32
3.2.3. Exemple avec des concepts des données 33
3.3. CADRE THÉORIQUE SPÉCIFIQUE : RBAC 35
3.3.1. Notation graphique, terminologie 37
3.4. CONCLUSION 41
CHAPITRE 4 : CONCEPTION DU SYTÈME 42
4.1. INRODUCTION 42
4.2. SPÉCIFICATION DES BESOINS 42
4.2.1. Besoins fonctionnels 42
4.2.2. Besoins non fonctionnels 44
4.2.3. Évaluation des contraintes non fonctionnelles 45
4.3. CONCEPTION DU SYSÈME 45
4.3.1. Architecture globale du système 46
4.1.1. Architecture logicielle (patron MVC) 47
4.1.2. Description des composantes principales 47
4.1.3. Interaction entre les modules 48
4.1.4. Présentation des outils et bibliothèques de programmation 49

4.1.5. Identification des acteurs 50
4.1.6. Diagramme de cas d'utilisation 51
4.1.7. Les Descriptions textuelles du cas d'utilisation 52
4.1.8. Diagramme de séquence 56
4.1.9. Diagramme de classe 57
- Relations et cardinalités 58
4.2. LANGAGE DE PROGRAMATION UTILISÉ 59
4.3. CONCLUSION 60
CHAPITRE 5 : REALISATION DU SYTÈME 61
5.1. INTRODUCTION 61
5.1.1. Système d'entités et canaux 61
5.1.1.1. Création et retrait d'entités 62
5.1.1.2. Création et retrait de canaux 64
5.1.2. Systèmes de sujets et objets sans rôles (DAC, MAC) 66
5.1.2.1. Création et retrait de sujet et objets 67
5.1.2.2. Création et retrait de canaux de lecture et écriture 71
5.1.2.3. Discrétionnaire d'Accès (DAC) 73
5.1.2.4. Modèle d'accès multi-niveaux (MAC) 75
5.1.3. Role-Based Access Control (RBAC) 87
5.1.3.1. Création, retrait de rôles, modification de rôles 88
5.1.3.2. La gestion d'attribution de rôles 93
5.1.4. Attribute-Based Access Control (ABAC) 98
5.1.5. Introduction de Sous-Systèmes Graphiques et de Boîtes Conteneurs 101
5.1.6. Simulation de temps d'exécution de l'algorithme 106
5.2. CONCLUSION 107
CHAPITRE 6 : CONCLUSION GÉNÉRALE 108
RÉFÉRENCES 115

LISTES DES FIGURES

Figure 1 : Niveaux d'habilitation 15
Figure 2 : Flux de données autorisés à l'aide de MLS 16
Figure 3 : Biba: Permissions et interdictions (extrait de [15]) 18
Figure 4 : Modèle de la Muraille de Chine (extrait de [1]) 19
Figure 5 : Noyau RBAC (extrait de [20]) 23
Figure 6: Un *canal* relation (a), ses classes d'équivalence étiquetées (b) et ses entités étiquetées (c). 33
Figure 7. a) Une table de rôles-autorisations, b) son graphe biparti correspondant, et c) son graphe étiqueté correspondant 39
Figure 8. a) Le graphe d'ordre partiel pour le réseau de la Fig.7 ; (b) un réseau équivalent ; c) sa table de rôles-autorisations 40
Figure 9: Architecture globale du système 46
Figure 10: Diagramme de cas d'utilisation 51
Figure 11: Diagramme de séquence 56
Figure 12: Diagramme de classe 58
Figure 13: Création Entité 62
Figure 14: Retrait d'entité 63
Figure 15: Création canal entre *E1* et *E2* 64
Figure 16: Retrait canal entre *E1* et *E2* 65
Figure 17: Interface d'accueil 67
Figure 18: Création d'entité (Sujet *S1*) 68
Figure 19: Ajout d'entité (objet *O1*) 69
Figure 20: Retrait de l'entité (*S1* et *O1*) 70
Figure 21: Ajout des canaux des permissions 71
Figure 22: Retrait du canal (*S1 R O1*) 72
Figure 23: Gestion de propriétaire 74
Figure 24: Ordre partiel de matrice de contrôle d'accès du Tableau 12(Option d'exécution Excel) 76
Figure 25: Flux de données des entités et leurs canaux du Tableau 12 et Figure 24 77
Figure 26: Exécution de commande de la Figure 24 du Tableau 12 78
Figure 27: Ordre partiel de matrice de contrôle d'accès du Tableau 13(Option d'exécution Excel) 80

Figure 28: Flux de données des entités et leurs canaux du Tableau 13 81
Figure 29: Interdiction globale 84
Figure 30: Interdiction ciblée(a) 85
Figure 31: Interdiction ciblée(b) 86
Figure 32:Gestion des rôles (*S1 R O1* avec le rôle *R1*) 88
Figure 33:Gestion des rôles (*S1 R*, *W O1* avec le rôle *R1*) 89
Figure 34:Retrait permission *W* au Rôle *R1* 90
Figure 35: Modification de la permission R sur l'objet O1 accordée au rôle R1 91
Figure 36:Permission Multiple 92
Figure 37 : Un réseau avec un rôle par sujet et Son ordre partiel 95
Figure 38: Exécution de commande de la Figure 37 du tableau 16 97
Figure 39:Gestion des attributs (ABAC) 100
Figure 40: Graphe principale (a) 103
Figure 41 : Sous graphe (b), (Composant connexe {*S3*, *S1*, *O2*, *O6*, *O8*}) 104
Figure 42: Sous graphe(c), (Composant connexe {*S6*, *S8*, *O3*, *O5*}) 105
Figure 43: Temps d'exécution de l'algorithme 106

LISTES DES TABLEAUX

Tableau 1 : Matrice de contrôle d'accès 9
Tableau 2 : Modèle de Lampson - Matrice de permissions 11
Tableau 3 : Évaluation des contraintes non fonctionnelles 45
Tableau 4: Cas d'utilisation 1 : Importer un fichier Excel 52
Tableau 5: Cas d'utilisation 2 : Saisir des commandes 53
Tableau 6: Cas d'utilisation 3 : Configurer les permissions 54
Tableau 7:Cas d'utilisation 4 : Gérer les rôles et les attributs 54
Tableau 8: Visualiser les relations et les étiquettes 55
Tableau 9:Acteurs et objets impliqués 57
Tableau 10: Commandes d'exécution (Entité) 62
Tableau 11: Commandes d'exécution (Sujet et Objet) 66
Tableau 12: Commande d'exécution (DAC) 73
Tableau 13: Matrice de contrôle d'accès de la Figure 24 et 25 75
Tableau 14: Listes des matrices contrôle d'accès de la Figure 27 79
Tableau 15: Commandes d'exécution (Interdiction sur les étiquettes) 83
Tableau 16: Commandes d'exécution (RBAC) 87
Tableau 17: Matrice de contrôle d'accès Rôle et Permissions de la Figure 37 93
Tableau 21: Liste des commandes implémentées (version française) 112
Tableau 24:Liste des commandes implémentées (version anglaise) 113
Tableau 25: Outils graphiques examinés 114

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ABAC	Contrôle d'accès basé sur les attributs
API	Application Programming Interface (Interface de programmation d'application)
CR	Peut lire
CW	Peut écrire
DAC	Contrôle d'accès discrétionnaire
IoT	Internet des objets
MAC	Contrôle d'accès obligatoire (Contrôle d'accès obligatoire)
MLS	Sécurité multi-niveaux
OBS	Objets (Objets)
OPS	Opérations (Opérations)
PA	Affectation des permissions
PRMS	Permissions (Autorisations)
RBAC	Contrôle d'accès basé sur les rôles
R	Rôle (Rôle)
SDN	Software-Defined Network (Réseau défini par logiciel)
SP	Permissions des sujets
SR	Rôles des sujets

SUBS	Sujets (Sujets)
UI	User Interface (Interface utilisateur)
Channel	Canal
UML	Unified Modeling Language
MVC	Modèle-Vue-Contrôleur

RÉSUMÉ

Dans les entreprises, la gestion des ressources sensibles et des flux d'information est encadrée par des systèmes de contrôle d'accès. Ces systèmes établissent un ensemble de règles définissant les permissions des utilisateurs en fonction des ressources et des modèles de sécurité employés. Ces règles peuvent être nombreuses et sont fréquemment ajustées en fonction des besoins opérationnels et des contraintes de sécurité.

Ce mémoire vise à montrer comment les concepts de sécurité des données, tels que le contrôle de flux d'information, la confidentialité et l'intégrité, peuvent être appliqués à plusieurs modèles de contrôle d'accès : le modèle discrétionnaire (DAC), le modèle multi-niveaux (MAC), le modèle basé sur les rôles (RBAC), et le modèle basé sur les attributs (ABAC). Nous développons un système graphique interactif capable d'exprimer et de modéliser chacun de ces paradigmes dans le cadre de la sécurité des données, renforçant ainsi la flexibilité et l'efficacité de la gestion des permissions et des flux de données.

Contrairement à la majorité des travaux qui reposent sur des modèles en treillis pour exprimer ces concepts, notre approche repose sur un cadre plus général, fondé sur la théorie des ordres partiels. Nous introduisons les notions d'ordres partiels, de classes d'équivalence et d'étiquettes de sécurité, qui permettent de modéliser les relations entre sujets et objets dans des systèmes de contrôle d'accès variés. Cela permet de régir les flux d'information entre eux et de définir leur niveau de confidentialité et d'intégrité.

Utilisant ces concepts, nous proposons une méthode permettant de dériver des configurations à partir des exigences de sécurité des données pour les différents modèles étudiés, y compris le DAC, MAC, RBAC et ABAC. Nous démontrons également que trois approches de définition des flux de données — par matrices de contrôle d'accès, par étiquetage de sécurité et par rôles — sont équivalentes du point de vue du flux de données qu'elles peuvent permettre et convertibles entre elles. Enfin, nous montrons comment les changements d'état ou les « reconfigurations » peuvent être formalisés dans ce cadre et allons analyser leur impact sur la sécurité des données.

Mots clés : IoT, DAC, MAC, Bell-LaPadula, ABAC, RBAC, contrôle d'accès, flux de données, sécurité des données, confidentialité, intégrité, contrôle à plusieurs niveaux, étiquetage de sécurité, reconfiguration.

ABSTRACT

In companies, the management of sensitive resources and information flows is framed by access control systems. These systems establish a set of rules defining user permissions according to the resources and security models used. These rules can be numerous and are frequently adjusted according to operational needs and security constraints.

This thesis aims to show how data security concepts, such as information flow control, confidentiality and integrity, can be applied to several access control models: the discretionary model (DAC), the mandatory model (MAC), the role-based model (RBAC), and the attribute-based model (ABAC). We develop an interactive graphical system capable of expressing and modeling each of these paradigms in the context of data security, thus reinforcing the flexibility and efficiency of permission and data flow management.

Unlike most works that rely on lattice models to express these concepts, our approach relies on a more general framework, based on partial order theory. We introduce the notions of partial orders, equivalence classes and security labels, which allow to model the relationships between subjects and objects in various access control systems. This allows to govern the information flows between them and to define their level of confidentiality and integrity.

By using these concepts, we propose a method to derive configurations from data security requirements for the different models studied, including DAC, MAC, RBAC, and ABAC We also demonstrate that three approaches to defining data flows—by access control matrices, by security labelling and by roles—are equivalent from the point of view of data flows allowed, and convertible between them. Finally, we show how state changes or “reconfigurations” can be formalized in this framework and analyze their impact on data security.

Keywords: IoT, DAC, MAC, Bell-LaPadula, ABAC, RBAC, access control, data flow, data security, confidentiality, integrity, multi-level control, security labeling, reconfiguration.

CHAPITRE 1

1.1. INTRODUCTION

Dans un monde de plus en plus interconnecté et numérique, la gestion des accès et des autorisations aux ressources numériques représente un enjeu crucial pour la sécurité des systèmes d'information. Avec l'énorme croissance du volume des données échangées et la complexité croissante des interactions entre utilisateurs et ressources, l'adoption de modèles performants et adaptatifs de contrôle d'accès devient indispensable pour répondre aux exigences de sécurité. Ces défis sont amplifiés dans des environnements dynamiques tels que l'Internet des Objets (IoT) [2] et le cloud computing, où les interactions et les changements de configuration peuvent être rapides et imprévisibles.

Traditionnellement, plusieurs modèles de contrôle d'accès ont été utilisés pour répondre à ces défis, chacun ayant ses forces et ses limites spécifiques. Le modèle discrétionnaire d'accès (DAC) permet aux propriétaires de ressources de gérer les autorisations selon leurs besoins [3], offrant ainsi une grande flexibilité, mais il peut souffrir de failles de sécurité en raison d'un manque de contrôle centralisé et de l'attribution manuelle des permissions. Le modèle Bell-LaPadula (MAC), quant à lui, se concentre sur la protection de la confidentialité [4] des informations dans des systèmes à niveaux, mais il reste rigide lorsqu'il s'agit de gérer des environnements modernes et distribués où la granularité des accès doit être fine et adaptable.

Face à ces limitations, le modèle de contrôle d'accès basé sur les rôles (RBAC) s'est imposé comme une solution plus adaptée pour les environnements complexes et évolutifs tels que l'IoT. Ce modèle simplifie la gestion des accès en attribuant des rôles aux utilisateurs, chaque rôle étant associé à un ensemble de permissions spécifiques[5], [6]. Cette approche permet de centraliser et de structurer les autorisations, réduisant ainsi le risque d'erreurs manuelles et d'incohérences dans la gestion des accès. Le modèle basé sur les attributs (ABAC) améliore cette flexibilité en attribuant des permissions en fonction des attributs des utilisateurs et des objets, mais il nécessite une gestion complexe des règles et des politiques.

Dans le cadre de ce projet, nous avons opté pour une solution hybride, intégrant plusieurs modèles de contrôle d’accès, notamment le DAC, MAC, RBAC et ABAC. Cette combinaison

permet d'achever une grande flexibilité dans et la gestion de la sécurité et des accès dans des systèmes complexes et distribués. Le modèle RBAC a été choisi comme cadre central en raison de sa capacité à s'adapter aux exigences modernes de gestion des autorisations, en particulier dans des contextes où les rôles et les permissions évoluent constamment. Par ailleurs, pour améliorer l'efficacité de la gestion des autorisations et des flux d'informations, nous avons intégré l'algorithme de Tarjan[7], qui permet d'identifier les composants fortement connectés dans un réseau d'entités, garantissant ainsi une gestion sécurisée et ordonnée des changements d'autorisations.

En outre, l'utilisation des réseaux bipartites dans la configuration RBAC de notre système permet de séparer les utilisateurs (sujets) des ressources (objets), assurant ainsi une gestion fine des opérations de lecture et d'écriture, quand cette distinction s'avère nécessaire. Cela nous permet d'analyser les flux d'information et de gérer l'impact des changements d'autorisations sur la sécurité des données. Enfin, pour éviter les conflits d'intérêts potentiels, nous intègrerons le modèle Brewer et Nash (également connu sous le nom de "Muraille de Chine"), qui permet de gérer les accès de manière stricte lorsque des conflits d'intérêts existent entre utilisateurs[1] et les deux autres modèles (DAC,ABAC).

Cette étude vise à démontrer que l'intégration de ces différents modèles de contrôle d'accès, combinée à l'algorithme de Tarjan et aux réseaux bipartites, offre une solution robuste et adaptable aux défis complexes de la gestion des accès et des flux d'information dans des systèmes modernes. En exploitant ces outils, nous cherchons à fournir une solution sécurisée et efficace qui améliore la gestion des permissions tout en garantissant la confidentialité, l'intégrité et la disponibilité des données échangées.

1.2. CONTEXTE ET PROBLÉMATIQUE

La gestion des accès et des autorisations aux ressources numériques constitue un enjeu majeur, en particulier dans des environnements complexes et dynamiques tels que l'Internet des Objets (IoT), le cloud computing, et les réseaux distribués. Ces systèmes intègrent des multitudes d'utilisateurs, d'objets, de services, ainsi que des relations évolutives entre ces entités. Garantir un contrôle d'accès strict et sécurisé est essentiel pour protéger la confidentialité, l'intégrité et

la disponibilité des données. Cependant, dans ce contexte, la gestion manuelle des autorisations devient rapidement inefficace et sujette à erreurs.

L'Internet des Objets, par exemple, illustre bien la complexité croissante de ces systèmes : un grand nombre d'appareils interagissent de manière autonome, créant des flux de données massifs et constants. Dans un tel environnement, le besoin de garantir que seules les entités autorisées puissent accéder aux ressources appropriées est primordial. Mais la gestion des autorisations, des rôles et des accès devient non seulement difficile à maintenir, mais aussi vulnérable aux erreurs humaines. Des décisions d'accès incorrectes peuvent entraîner des violations de sécurité, compromettant ainsi la confidentialité et l'intégrité des données.

Traditionnellement, des modèles tels que le DAC (Discretionary Access Control) sera utilisé pour gérer les accès en donnant aux propriétaires des ressources le contrôle total sur les autorisations. Cependant, ce modèle montre des limites importantes dans des systèmes à grande échelle ou des environnements distribués où les besoins en gestion centralisée et cohérente des accès sont critiques. Le modèle MAC, particulièrement illustré par le modèle Bell-LaPadula, a permis de structurer les accès selon des niveaux de sécurité et de confidentialité [4], mais il reste limité dans sa flexibilité et son adaptabilité dans des contextes où les rôles et les autorisations évoluent dynamiquement, Notre approche vient améliorer tous ces aspects.

Face à ces défis, le modèle RBAC (Role-Based Access Control) a émergé comme une solution plus adaptée aux environnements modernes. Il simplifie la gestion des accès en attribuant des rôles aux utilisateurs, chaque rôle étant associé à des autorisations spécifiques sur des ressources. Dans des nombreuses variations et adaptations, RBAC est utilisé dans ce nombreux organisme et systèmes. Avec des extensions, son utilisation est envisagée dans le cloud et dans l’Internet des objets [2], [8], [9]. De plus, la capacité du modèle RBAC à gérer les flux d'informations et les autorisations de manière plus ordonnée en fait un choix idéal pour des systèmes où les autorisations doivent constamment être modifiées en fonction des changements d'accès.

Le modèle ABAC, quant à lui, offre plus de granularité en attribuant des autorisations en fonction d'attributs spécifiques des entités, mais il complexifie la gestion des politiques d'accès.

La problématique principale de ce projet consiste donc à concevoir et mettre en œuvre un système de gestion des accès capable d'intégrer ces différents modèles de contrôle d'accès (DAC, MAC, RBAC et ABAC), tout en assurant une gestion efficace des flux d'information et des permissions dans un environnement dynamique. L'objectif est de réduire les erreurs humaines, améliorer les autorisations, et de garantir la sécurité des données tout en tenant compte des besoins spécifiques de chaque modèle. La complexité et la diversité des systèmes modernes rendent nécessaire une approche hybride et adaptable pour répondre aux exigences croissantes en matière de sécurité et de gestion des accès.

1.3. L'IMPORTANCE DE LA VISUALISATION DES REALATIONS ENTITÉS-PERMISSIONS

La visualisation des relations entre les entités et leurs permissions est un atout crucial pour les administrateurs de systèmes complexes. Elle offre une vue d'ensemble claire des interactions entre les utilisateurs, les rôles, et les ressources, permettant ainsi d'identifier rapidement les incohérences ou vulnérabilités. En matérialisant graphiquement ces relations, les administrateurs peuvent mieux comprendre les autorisations en place et ajuster les permissions ou les rôles de manière plus proactive et précise.

1.4. OBJECTIF PRINCIPALE

L'objectif principal de ce projet est de concevoir et développer un système graphique interactif capable de gérer et de visualiser les relations entités-permissions dans un cadre qui englobe les modèles DAC, MAC, RBAC et ABAC. Ce système doit être flexible pour s'adapter aux exigences des environnements complexes, en particulier ceux impliquant des appareils connectés comme par exemple l'IoT. Il doit fournir une visualisation précise des rôles, des entités et des autorisations afin d'améliorer la sécurité et garantir la confidentialité et l'intégrité des données. Cet outil servira comme un outil d'expérimentation pour les différents utilisateurs.

1.5. OBJECTIF SPÉCIFIQUE

Les objectifs spécifiques du projet incluent :

- Développer une interface utilisateur intuitive permettant l'ajout, la suppression et la modification d'entités et de permissions, en tenant compte des spécificités de chaque modèle de contrôle d'accès (DAC, MAC, RBAC, ABAC).

- Implémenter un cadre capable de gérer la diversité des modèles de contrôle d'accès tout en intégrant une approche basée sur les rôles (RBAC) pour simplifier la gestion des permissions.

- Intégrer des outils de visualisation (comme NetworkX et PyVis) pour représenter graphiquement les relations entités-permissions et permettre une meilleure compréhension des flux d'information et des accès.

- Réduire les redondances et améliorer les relations permissions-entités à l'aide d'algorithmes comme l'algorithme de Tarjan, permettant de repérer les composants fortement connexes [7] et de structurer les rôles et autorisations dans un cadre ordonné et sécurisé.

1.6. CONTRIBUTION VISÉE

Ce projet se veut une contribution significative à la gestion des accès et des autorisations dans des environnements complexes, en particulier ceux impliquant l'IoT, en combinant les forces de plusieurs modèles de contrôle d'accès avec des outils de visualisation performants. Cette combinaison permet non seulement de sécuriser les systèmes, mais aussi d'améliorer la gestion des autorisations de manière claire, intuitive et efficace. De plus, le système offrira deux méthodes d'interaction :

1. **L'option fichier Excel** permet un traitement rapide et automatisé des données existantes, facilitant l'importation et la gestion de grandes quantités d'informations.

2. **L’option commandes manuelles** propose une manipulation dynamique, offrant plus de flexibilité pour ajuster les autorisations et les configurations selon les besoins spécifiques.

1.7. ORGANISATION DU MÉMOIRE

Ce mémoire est structuré en six chapitres principaux afin de présenter de manière logique et progressive les différentes étapes de notre projet. Il comprend les sections suivantes :

Chapitre 1 : Introduction

Ce chapitre introduit le sujet du contrôle d'accès, le contexte dans lequel ils s'inscrivent, ainsi que les enjeux qu'ils soulèvent. Il souligne l'importance de la gestion des autorisations dans des systèmes complexes et examine les défis actuels liés à la gestion des accès et des autorisations. L'accent est également mis sur la nécessité de visualiser les relations entre entités et autorisations. En outre, ce chapitre expose les objectifs de notre projet ainsi que les motivations qui l'ont inspiré, de vérification des politiques de sécurité, développé dans le cadre de ce mémoire. Enfin, il présente la contribution attendue de ce travail ainsi que l'organisation générale du mémoire.

Chapitre 2 : Revue de la Littérature

Ce chapitre explore les modèles existants de gestion des accès. Il introduit ensuite certains des modèles de contrôle d'accès les plus réputés dans la littérature, en commençant par le contrôle d’accès discrétionnaire, avec notamment le modèle de Lampson et celui de Harrison-Ruzzo-Ullmann, le chapitre présente les modèles de contrôle d’accès à plusieurs niveaux (MAC), tels que le modèle Bell-LaPadula et Biba, le modèle de contrôle d'accès basé sur les rôles (RBAC). Enfin le modèle basé sur les attributs (ABAC).

Chapitre 3 : Cadre théorique

Ce chapitre décrit la théorie que nous avons utilisé pour développer notre approche.

Chapitre 4 : Conception du système

Le chapitre sur la conception du système décrit l'architecture modulaire organisée en trois couches : entrée utilisateur, traitement et visualisation.

Il présente l'intégration des modèles DAC, MAC, RBAC et ABAC pour gérer les permissions et les flux d'information.

Les diagrammes UML (cas d'utilisation, classes, séquence, architecture) illustrent les interactions et la structure des composants.

La conception permet la configuration dynamique des entités et la vérification des règles de sécurité.

Des interfaces interactives facilitent l'importation de données et la saisie de commandes. Ce chapitre constitue la base technique pour l'implémentation et la validation du système.

Chapitre 5 : Réalisation du système

Ce chapitre présente le développement de notre système, en présentant son choix technologique, ainsi que les options d'exécution via le fichier Excel et les commandes manuelles. Des exemples concrets et des visualisations graphiques illustrent les fonctionnalités du système.

Chapitre 6 : Conclusion générale

Ce chapitre synthétise tout le travail réalisé dans le cadre de ce projet. Il résume les contributions majeures, les résultats obtenus et les enseignements tirés de l'implémentation des différents modèles de contrôle d'accès. La conclusion met en lumière l'impact du projet sur la gestion des autorisations dans les systèmes complexes et propose une réflexion sur les avancées possibles dans ce domaine.

CHAPITRE 2 : REVUE DE LA LITÉRATURE

2.1. INTRODUCTION

La gestion des accès dans les systèmes d'information complexes est un domaine clé dans le maintien de la sécurité des données. Plusieurs modèles de contrôle d'accès ont été développés au fil du temps pour répondre à des exigences spécifiques de confidentialité, d'intégrité, et d'efficacité. Parmi ces modèles, on retrouve le Discretionary Access Control (DAC), le Mandatory Access Control (MAC), le Role-Based Access Control (RBAC) et le Attribute-Based Access Control (ABAC). Ces modèles constituent la base théorique sur laquelle reposent de nombreuses implémentations actuelles de systèmes de gestion des accès, et ils répondent chacun à des besoins distincts en matière de sécurité.

Dans cette section, nous allons examiner en détail ces différents modèles, en nous concentrant sur leurs mécanismes, leurs forces et leurs faiblesses, ainsi que sur leur pertinence dans des environnements modernes tels que l'Internet des Objets (IoT). Nous analyserons également la manière dont ces modèles peuvent être intégrés ou combinés pour augmenter la sécurité dans des systèmes complexes et évolutifs. Il est important de noter que ces modèles ont beaucoup d'utilisations, mais nous allons nous concentrer sur leur utilisation pour la protection des données.

2.2. MODÈLE D'ACCÈS ET SÉCURITÉ

Les modèles de contrôle d'accès sont utilisés pour définir qui peut accéder à quelles ressources et dans quelles conditions. Ces modèles ont évolué pour répondre aux besoins croissants de sécurité dans des environnements variés, allant des réseaux d'entreprise traditionnels aux systèmes IoT hautement distribués. Nous allons examiner ici les principaux modèles : DAC, MAC, RBAC et ABAC, mais avant tout parlons de modèle de matrice de contrôle d'accès.

Dans sa forme la plus fondamentale, une matrice de contrôle d'accès est une matrice booléenne qui, pour chaque paire d'entités e_i et e_j, contient la valeur *1* si des données peuvent

être transférés de e_i à e_i, ou *0* sinon. Dans le premier cas, nous dirons qu'il y a un *canal* de e_i à e_i. Dans la théorie du contrôle d'accès aux données, quand il y a distinction entre sujets et objets, cette matrice est souvent exprimée définissant une fonction $A : S \times O \longrightarrow 2^R$ qui donne pour chaque couple (sujet, objet), l'ensemble des droits que le sujet a sur l'objet [10],[11]. Souvent les droits considérés peuvent être des droits de suppression, modification, etc., mais dans ce travail nous nous limiterons à considérer les droits de lecture et écriture.

L'analyse s'appuie sur des états spécifiques, appelés états de protection, représentés sous forme de triplets *(S, O, A)* où *S* désigne l'ensemble des sujets, *O* l'ensemble des objets et *A* la matrice de contrôle d'accès. Lorsqu'un sujet, un objet ou un droit d'accès est ajouté ou supprimé, l'état du système change.

	Objet 1 (O1)	Objet 2 (O2)	Objet 3 (O3)	Objet 4 (O4)
Sujet1 (S1)	Écrire		Lire	
Sujet1 (S2)		Lire		Écrire
Sujet1 (S3)	Lire	Écrire		Lire
Sujet1 (S4)	Lire, Écrire		Lire	

Tableau 1 : Matrice de contrôle d'accès

Les droits d'accès lire et écrire associés au couple (*S1, O1*), (*S1, O3*) dans le tableau 1 qui représente l'association d'un sujet *S1* et d'un objet *O1* et *O3*. Cela signifie que *S1* peut donc accéder en lecture à l'objet *O3* et en écriture à l'objet *O1*.

Cette atructure fondamentale ne prévoit pas, en elle-même, ni des mécanismes pour la modifier, ni des méthodes de représentation qui permettent d'exprimer aisément les politique d'entreprise. Pour cela, les modèles suivants ont été développés.

2.2.1. Modèle Discrétionnaire d'Accès (DAC)

Le Contrôle d'Accès Discrétionnaire (DAC) , est l'un des premiers modèles de contrôle d'accès. Il est basé sur l'idée, conforme aux principes de Unix-Linux, que chaque ressource (ou objet) a un *propriétaire*, qui normalement est l'utilisateur qui a créé la ressource. Il permet aux propriétaires de ressources de définir des règles d'accès pour les autres utilisateurs, généralement sous la forme<Sujet, Objet, Action>, ce qui signifie qu'un sujet (comme un utilisateur ou un processus) peut effectuer une action spécifique (lire, écrire, ou modifier) sur un objet donné (qui pourrait être un fichier ou un programme, etc.). La décision d'accès repose généralement sur une autorisation qui est déterminée à partir des informations recueillies lors de l'authentification (comme le nom d'utilisateur et le mot de passe). Dans la plupart des modèles DAC, le propriétaire de la ressource a la capacité de modifier les autorisations à sa discrétion, d'où l'origine du nom « contrôle d'accès discrétionnaire ». Le modèle DAC connaît plusieurs modèles, tel que le modèle de Lampson [10],[3, 12], que nous aborderons par la suite.

2.2.1.1. Modèle de Lampson

Le modèle de Lampson [10], proposé en 1971, repose sur la notion de matrice de contrôle d'accès, où chaque élément de la matrice correspond aux permissions d'un sujet sur un objet donné . Les sujets *s* (utilisateurs ou processus) et les objets *o* (fichiers, programmes) sont représentés respectivement par les lignes et les colonnes de la matrice, comme nous avons vu précédemment. Les cellules de cette matrice contiennent les actions que les sujets peuvent effectuer sur les objets, telles que la lecture ou l'écriture. Bien que le modèle de Lampson offre une vision claire des autorisations, sa représentation matricielle est peu flexible pour des systèmes avec de nombreuses entités, car elle peut devenir complexe à manipuler. Il ne permet pas d'implémenter des politiques définies au niveau de l'entreprise.

	Objet 1(Fichier A)	Objet 2(Fichier B)	Objet 3(Fichier C)
Sujet1 (Luigi)	Lire, Écrire, Propriétaire	Lire	Propriétaire
Sujet2 (Ripama)		Lire, Écrire	Lire
Sujet3 (Waara)	Lire	Propriétaire	Lire, Écrire

Tableau 2 : Modèle de Lampson - Matrice de permissions

Le modèle de Lampson utilise une matrice pour représenter les autorisations d'accès entre les sujets *s* (utilisateurs ou processus) et les objets *o* (fichiers, bases de données, etc.). Chaque cellule de cette matrice définit les actions autorisées pour un sujet *s* donné sur un objet *o* donné.

- Ligne (Sujets) : Représente les utilisateurs ou les processus qui interagissent avec les objets.

- Colonnes (Objets) : Représentent les ressources auxquelles les sujets peuvent accéder.

- Cellules (Permissions) : Indiquent les types d'accès autorisés pour un sujet sur un objet donné (par exemple, "Lire", "Écrire").

Explications :

- Luigi est propriétaire de « Fichier A » et « Fichier C », ce qui signifie qu'il peut attribuer ou retirer des autorisations sur ces fichiers, il a aussi la permission de lire « Fichier B ».
- Ripama a l'autorisation de lire et écrire sur « Fichier B », et uniquement lire « Fichier C ».
- Waara est propriétaire de « Fichier B », à la permission de lire « Fichier A », et peut lire, écrire sur « Fichier C ».

Cela permet de visualiser clairement qui a accès à quoi et quels types de permissions sont attribués, formant ainsi la base de la matrice de contrôle d'accès du modèle de Lampson.

Le modèle de Lampson établit les règles suivantes [12] :

- **Règle 1** : Le propriétaire d'un objet peut ajouter des attributs d'accès à cet objet pour d'autres utilisateurs.

- **Règle 2** : Un utilisateur disposant du droit de délégation peut accorder ses propres attributs d'accès à un objet donné à un autre utilisateur.

- **Règle 3** : Un utilisateur peut retirer les droits d'accès dans les domaines sur lesquels il exerce un contrôle.

- **Règle 4** : Le propriétaire d'un objet a le droit de supprimer les attributs d'accès associés à cet objet.

Ces tableaux permettent de capturer les différentes autorisations que les sujets ont sur les objets à un état donné et de montrer les droits du propriétaire.

2.2.1.2. Conclusion dur les modèles discrétionnaires

Le modèle Discrétionnaire d'Accès (DAC) est l'un des mécanismes les plus courants pour gérer les permissions dans les systèmes informatiques. Il fonctionne en permettant aux propriétaires de ressources de contrôler directement les accès aux objets, à travers des listes de contrôle d'accès ou des matrices de permissions, ce qui rend la gestion des autorisations flexible et granulaire.

Cependant, les modèles discrétionnaires présentent certains problèmes de sécurité. En effet, comme l'attribution des droits d'accès est confiée aux sujets, il est possible qu'un sujet accorde des droits "sensibles" à des sujets malveillants. Dans de tels cas, un sujet malveillant peut compromettre des informations cruciales ou accéder à des contenus auxquels il ne devrait pas avoir accès comme l'exemple des chevaux de Troie. Le modèle HRU a été créé pour souligner ces faiblesses et montrer que le problème de la fuite des droits est indécidable en principe, c.à.d. qu'il n'est pas possible d'écrire un programme qui, en général, puisse détecter toutes les

vulnérabilités existantes dans un modèle de ce type mais notre système ne traite pas ce modèle c'est pour cela que nous ne rentrons pas en détaille concernant ce modèle.

2.2.2. Modèle d'accès multi-niveaux (MAC)

Dans les théories traditionnelles de la sécurité informatique, les modèles MAC sont souvent décrits comme des modèles multi-niveaux, où les entités (sujets et objets) sont classées selon différents niveaux de sécurité ou d'intégrité. Contrairement aux modèles DAC, dans les MAC, l'attribution des droits d'accès n'est pas laissée aux sujets de la même façon.

2.2.2.1. Modèle de Bell-LaPadula

Le modèle de Bell-LaPadula [4] est un modèle de sécurité informatique axé sur le maintien de la confidentialité des informations au sein d'un système sécurisé. Il a été développé dans les années 1970 par David Elliott Bell et Leonard J. LaPadula, et il reste l'un des modèles les plus influents dans le domaine de la sécurité des systèmes d'information. Ce modèle repose sur une organisation des sujets et des objets selon des niveaux de sécurité, établis en fonction de leur importance dans le système de protection des données. Par exemple, ces niveaux peuvent inclure : *confidentiel*, *secret* et *top secret*. Ces niveaux sont ordonnés selon une relation d'ordre (≤), avec une hiérarchie qui peut être représentée comme suit : $confidentiel \leq secret \leq top\ secret$ [13] .

Ce modèle se base sur des concepts fondamentaux comme suit [13]:

Le niveau d'habilitation des sujets, chaque sujet est associé à un niveau de sécurité spécifique, appelé niveau d'habilitation, défini par une fonction : $f_S : S \rightarrow L$, où S est l'ensemble des sujets et L, l'ensemble des niveaux de sécurité.

Le niveau de classification des objets : de manière similaire, chaque objet est assigné à un niveau de sécurité appelé niveau de classification. Cela est formalisé par une fonction

$f_S : O \rightarrow L$, où O est l'ensemble des objets. Les autorisations d'accès d'un sujet à un objet dans ce modèle dépendent de deux règles fondamentales appelées propriétés de sécurité.

- Propriété simple (*Simple Security Property*) : Elle stipule qu'un sujet **s** peut lire un objet **o** uniquement si le niveau d'habilitation du sujet est supérieur ou égal au niveau de classification de l'objet. Cette règle est formalisée par :
  $\forall s \in S, \forall o \in O, (s, o, read) \in M \Rightarrow fs(s) \geq fo(o)$

  Cette propriété garantit la confidentialité des informations en empêchant un sujet d'accéder à des données classées à un niveau supérieur à son habilitation.
- Propriété étoile (*Star Property*) : Cette propriété indique qu'un sujet o peut écrire dans un objet s uniquement si le niveau d'habilitation du sujet est inférieur ou égal au niveau de classification de l'objet. Formellement :
  $\forall s \in S, \forall o \in O, (s, o, write) \in M \Rightarrow fs(s) \leq fo(o)$
  Cette règle, souvent résumée par « pas d'écriture vers un niveau inférieur », vise à empêcher la fuite d'informations sensibles vers des niveaux de sécurité inférieurs, notamment en cas d'attaques comme les chevaux de Troie.

  Un cheval de Troie est un programme malveillant qui effectue certaines actions sans que l'utilisateur qui l'a lancé en ait conscience, utilisant ainsi les privilèges de cet utilisateur pour détourner des informations. Pour empêcher un cheval de Troie de transmettre des informations classées « top secret » dans un document de niveau " confidentiel" ($confidentiel \leq secret \leq top\ secret$), cette écriture "vers le bas" est interdite. Le modèle assure ainsi la préservation de la confidentialité des données, à condition que ces dernières soient placées à des niveau de secret appropriés.

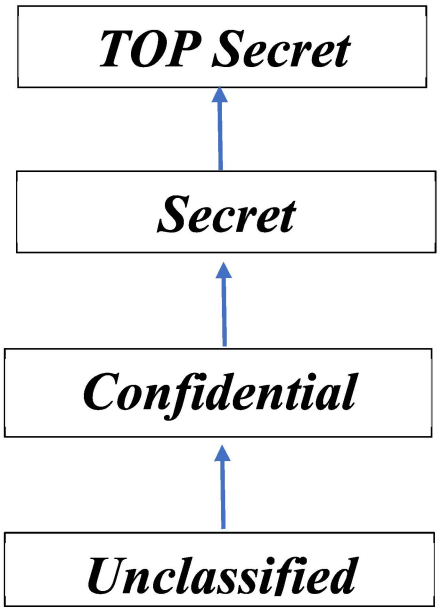


Figure 1 : Niveaux d'habilitation

La figure 1, intitulée « Niveaux d'habilitation », illustre les niveaux d'habilitation qui furent initialement établis par la communauté de la défense américaine. Seuls les utilisateurs disposant d'une habilitation de niveau confidentiel peuvent accéder aux documents classés dans cette catégorie. Toutefois, ces utilisateurs n'ont pas le droit de consulter des documents nécessitant des habilitations ou des niveaux de sécurité supérieurs. Ils peuvent uniquement lire les documents ayant des niveaux d'habilitation inférieurs et écrire sur ceux ayant des niveaux d'habilitation plus élevés.

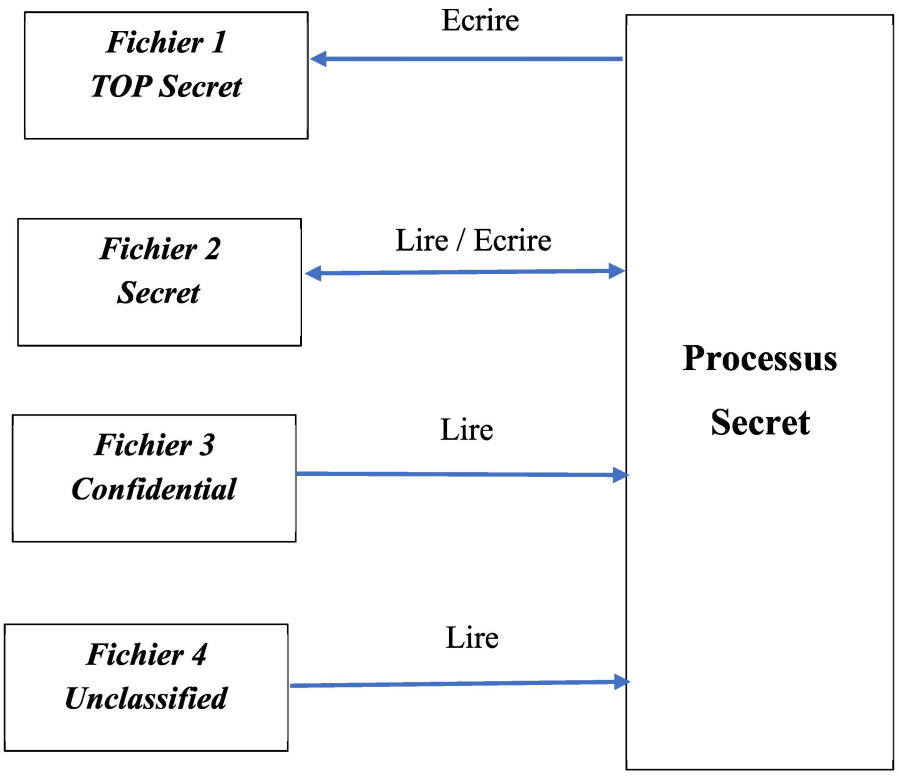


Figure 2 : Flux de données autorisés à l'aide de MLS

La figure 2, intitulée « Flux de données autorisés avec MLS », illustre tous les échanges de données permis entre un sujet fonctionnant sous le niveau de sécurité « Secret » et divers objets ayant des niveaux de sécurité variés.

2.2.2.2. Modèle d'intégrité de Biba

Le modèle d'intégrité de Biba est un modèle de sécurité informatique axé sur le maintien de l'intégrité des données. Contrairement au modèle de Bell-LaPadula, qui met l'accent sur la confidentialité ou secret, le modèle de Biba vise à garantir que les informations au sein d'un système ne sont ni corrompues ni modifiées de manière non autorisée. Ce modèle a été développé par Kenneth J. Biba en 1977 [14].

Les propriétés de sécurité dans le modèle de Biba se démontre comme [13] :

- Propriété simple

  Cette propriété, dans le modèle de Biba, stipule qu'un sujet s peut lire un objet o uniquement si son niveau d'habilitation est inférieur ou égal au niveau de classification de l'objet. Cette règle est formalisée comme suit :

  $$\forall s \in S,\ \forall o \in O,\ (s, o, read) \in M \Rightarrow fs(s) \leq fo(o).$$

  Ici, S représente l'ensemble des sujets, O l'ensemble des objets, fs la fonction d'habilitation, et fo la fonction de classification. Cette propriété vise à empêcher un utilisateur ayant un faible niveau d'intégrité de consulter des données critiques, préservant ainsi leur fiabilité.

- Propriété étoile

  La propriété étoile redéfinie dans ce modèle précise qu'un sujet **s** peut écrire dans un objet o uniquement si son niveau d'habilitation est supérieur ou égal au niveau de classification de l'objet :

  $$\forall s \in S,\ \forall o \in O,\ (s, o, read) \in M \Rightarrow fs(s) \geq fo(o).$$

  Cette règle empêche qu'un sujet ayant une habilitation élevée écrive dans des objets ayant un niveau d'intégrité inférieur, ce qui pourrait compromettre l'intégrité de ces informations.

Ces deux propriétés se résument souvent de manière concise par les expressions :

- « Pas de lecture depuis un niveau inférieur. »

- « Pas d'écriture vers un niveau supérieur. »

Dans un système où les niveaux d'intégrité sont hiérarchisés (*confidentiel* ≤ *secret* ≤ *top secret*), cela signifie qu'un utilisateur habilité au niveau *secret* ne peut pas modifier des informations classées *top secret*. Cette restriction protège les données sensibles contre des modifications malveillantes ou accidentelles, assurant ainsi leur intégrité.

Par exemple, un sujet malveillant disposant d'une habilitation *secrète* ne pourra pas introduire de fausses informations dans un document *top secret*. En revanche, ce sujet pourra consulter les informations classées *top secret* mais n'aura pas accès à des objets classés à un niveau inférieur comme *confidentiel*.

Le modèle de Biba s'assure ainsi que, même en cas d'attaque, les données critiques demeurent fiables et protégées contre toute manipulation non autorisée.

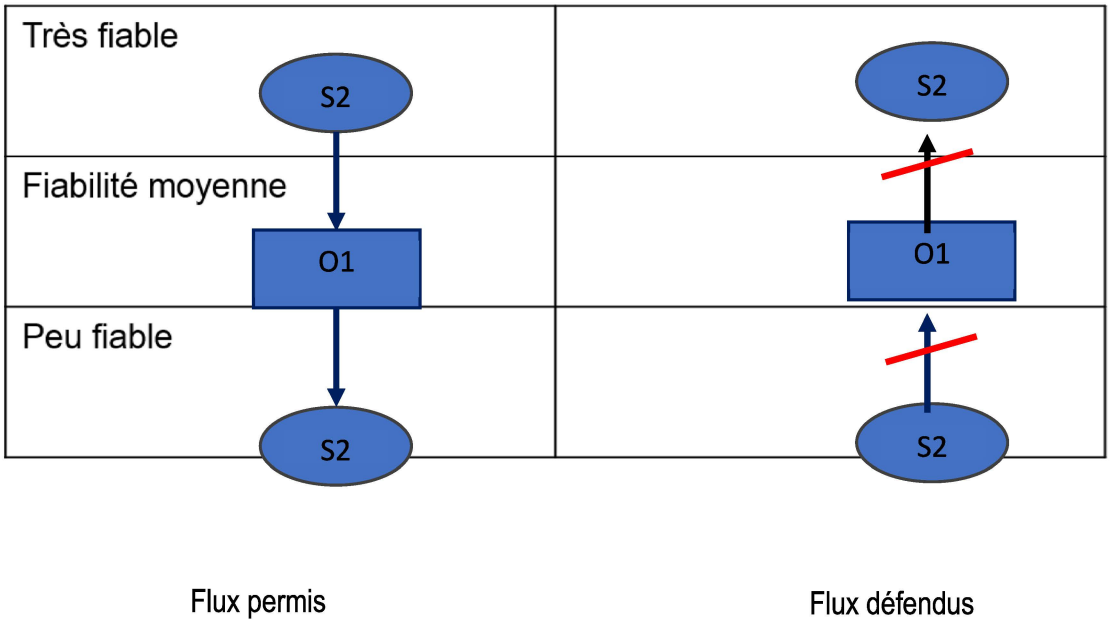


Figure 3 : Biba: Permissions et interdictions (extrait de [15])

Un sujet ayant un certain niveau d'intégrité ne peut pas invoquer ou initier un certain droit d'un niveau supérieur. Cette règle empêche une escalade de privilèges qui pourrait nuire à la sécurité du système [14].

Le modèle de Biba est souvent utilisé en complément du modèle de Bell-LaPadula, permettant ainsi d'assurer à la fois la confidentialité et l'intégrité des données dans un système informatique. Par exemple, dans des contextes militaires ou gouvernementaux, l'utilisation simultanée des deux modèles permet de garantir que les informations sensibles restent à la fois protégées contre les accès non autorisés et gardés dans leur état d'origine sans corruption ou altération. Cependant, Sandhu [16] a justement observé que les niveaux de confidentialité et de secret

peuvent être représentés dans un seul ordre partiel, et celle-ci sera l'approche qui sera adoptée dans notre recherche.

2.2.2.3. Modèle de Brewer et Nash

Le modèle de Brewer et Nash, également connu sous le nom de modèle " Muraille de Chine " [1], est un modèle de sécurité de contrôle d'accès élaboré pour protéger les informations sensibles contre les conflits d'intérêts, particulièrement dans des environnements où des données confidentielles de plusieurs clients ou parties sont gérées simultanément par une organisation. Ce modèle a été introduit par David FC Brewer et Michael J. Nash en 1989 [1].
L'objectif de ce modèle est de s'assurer qu'un utilisateur ne puisse pas accéder simultanément à des données sensibles de différentes parties concurrentes. Cela est particulièrement pertinent dans des secteurs tels que la finance, la consultance, le conseil juridique ou toute autre activité où des informations concurrentielles doivent être séparées pour éviter des abus ou des conflits d'intérêts.
La « Muraille de Chine » est en principe une barrière éthique entre différentes informations concurrentes. Les données sont classées en conflits d'intérêts, de sorte que lorsqu'un utilisateur accède à des données dans une classe spécifique, il ne peut plus accéder à d'autres classes qui présentent des conflits potentiels.

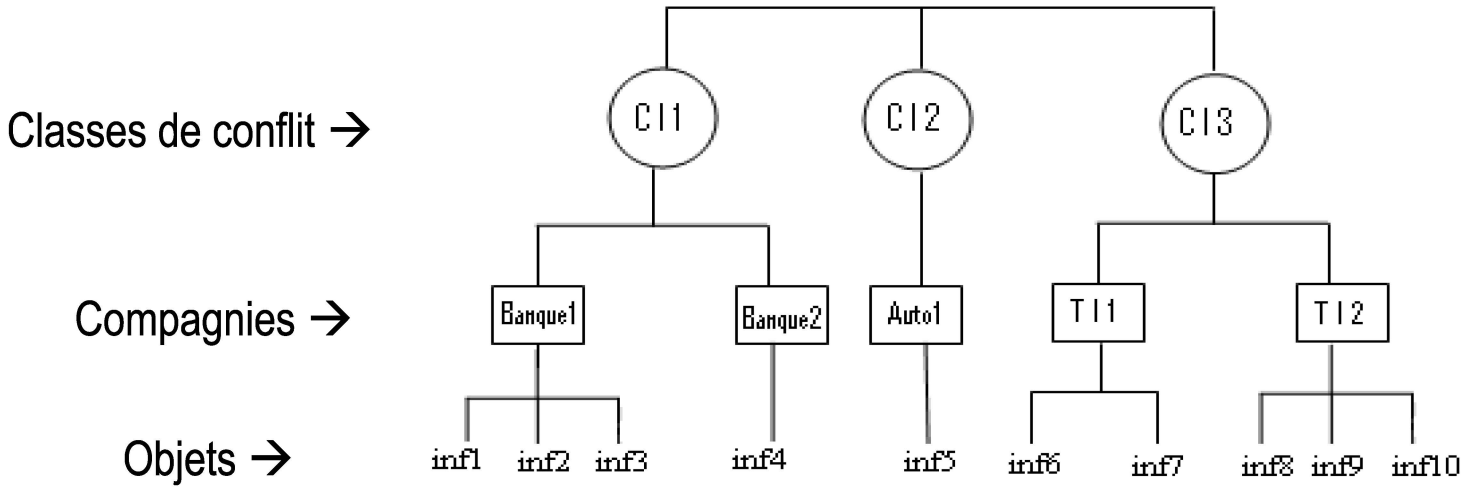


Figure 4 : Modèle de la Muraille de Chine (extrait de [1])

L'application des propriétés du modèle sont :

- Propriété simple

  Un sujet *s* peut accéder à un objet *o* (lecture) si :

  - *s* n'a pas encore accédé à des objets appartenant à une autre compagnie dans la même classe de conflit.
  - Ou *s* accède uniquement à des objets de la même compagnie.

  Prenons un exemple :

  Dans la figure 4, un sujet ayant accédé à *inf1(Banque1)* dans la classe *C11* ne pourra pas accéder à inf4(Banque2) car cela introduirait un conflit d'intérêts.

- Propriété étoile

  Un sujet *s* peut écrire dans un objet *o* si :

  - *s* a le droit de lire cet objet.
  - *s* n'introduit pas de données provenant d'une autre compagnie ou d'une autre classe de conflit.

  Exemple **:**

  Dans la figure 4, un sujet travaillant sur *inf5 (Auto1, classe C12)* ne pourra pas écrire des informations provenant de *inf3 (Banque1, classe C11)* dans cet objet. Cela préserve la séparation stricte des données entre les classes.

Prenons un exemple d'utilisation pratique dans un cabinet de conseil :
Si un employé travaille avec les données de *Banque1 (inf1, inf2, inf3)* dans la classe *C11*, il sera empêché d'accéder ou de manipuler les informations de *Banque2 (inf4)* dans la même classe *C11*. Cela évite que des données sensibles de deux concurrents bancaires soient mélangées ou utilisées de manière inappropriée.
De manière similaire, un consultant travaillant pour Auto1 dans C12 ne pourra pas écrire ou partager des informations de C11 (Banque1 ou Banque2) ou de C13(T11 ou T12).

Nous verrons que le modèle de la Muraille de Chine est implémenté dans notre méthode par la prohibition de certaines étiquettes.

2.2.2.4. Conclusion sur le modèle d'accès multi-niveaux

Le modèle MAC englobe plusieurs sous-modèles, chacun apportant des solutions spécifiques pour renforcer la sécurité des systèmes. Le système à niveaux, tel que le modèle Bell-LaPadula, met l'accent sur la confidentialité des informations [4] en imposant des restrictions sur les flux de données entre différents niveaux de classification. Il garantit qu'aucune information classée ne peut être transmise à un niveau inférieur, préservant ainsi la confidentialité [4]. En complément, le modèle d'intégrité de Biba [14]se concentre sur la protection de l'intégrité des données en empêchant les utilisateurs à des niveaux inférieurs de modifier des informations critiques, garantissant ainsi que seules les sources fiables peuvent effectuer des modifications. Quant au modèle de Brewer et Nash[1],[17] (ou modèle de la Muraille de Chine), il vise à éviter les conflits d'intérêts dans les environnements commerciaux en contrôlant l'accès en fonction du contexte, notamment lorsqu'il s'agit de clients concurrents. Ce modèle est souvent utilisé dans des situations où la neutralité des consultants ou des agents est essentielle.

En somme, chacun de ces modèles MAC joue un rôle clé dans des environnements nécessitant des contrôles d'accès rigoureux, en fonction de critères de confidentialité, d'intégrité ou de conflits d'intérêts. Le choix d'un modèle dépend largement des objectifs de sécurité spécifiques d'un système. Nous verrons plus tard que le modèle que nous proposons prend en charge tous ces besoins dans un cadre unifié.

2.2.3. Modèle RBAC

Le modèle RBAC (Role-Based Access Control), ou contrôle d'accès basé sur les rôles [18], est l'un des modèles de contrôle d'accès utilisés dans les systèmes d'information modernes. Ce modèle associe les autorisations aux rôles plutôt qu'aux utilisateurs individuels, ce qui simplifie la gestion des autorisations dans des environnements complexes. Dans un système RBAC, les utilisateurs ne reçoivent pas directement des autorisations sur les ressources, mais se voient plutôt attribuer des rôles spécifiques, qui eux-mêmes sont liés à des ensembles de permissions [18].

Dans le modèle RBAC, les rôles sont définis par l'administrateur en fonction des responsabilités et des autorités généralement stables au sein de la gestion, et chaque rôle est associé à des

fonctions spécifiques. Un utilisateur acquiert les permissions associées à un rôle en assumant rôle. Dès qu'un utilisateur sera affecté à des rôles, il pourra ouvrir des sessions utilisant des sous-ensembles de ses rôles. Il recevra alors les permissions et pourra exécuter les tâches qui lui sont associées. L'attribution des permissions directement aux rôles, plutôt qu'aux utilisateurs, permet une grande flexibilité et une granularité fine de l'attribution des autorisations [19].

2.2.3.1. Principes du modèle RBAC

Le modèle RBAC est caractérisé par :

- Rôles : Un rôle donne accès à une collection de permissions. Par exemple, un administrateur système peut avoir un rôle qui lui permet de lire, écrire et supprimer des fichiers, tandis qu'un utilisateur standard peut seulement avoir la permission de lire certains fichiers.

- Utilisateurs : Les utilisateurs représentent les individus ou entités qui utilisent le système. Dans le modèle RBAC, les utilisateurs se voient assigner un ou plusieurs rôles qui lui donnent accès aux permissions moyennant l'ouverture de sessions.

- Permissions : Les permissions définissent les actions qui peuvent être effectuées sur des ressources (lire, écrire, exécuter, etc.).

- Sessions : Une session permet à un utilisateur d'activer certains rôles, avec leurs permissions, lors de son activation.

Dans les systèmes complexes, RBAC facilite la gestion des permissions. En assignant des rôles plutôt que des permissions individuelles, l'administration devient plus simple, plus fluide et moins sujette aux erreurs [19]. Cela permet une meilleure cohérence dans l'application des politiques de sécurité.

Dans les environnements où de nombreux utilisateurs interagissent avec des multitudes de ressources, RBAC réduit la complexité. Il permet de regrouper les permissions par rôle et de

gérer les utilisateurs par groupe, plutôt que de gérer les autorisations individuellement. Une des forces du RBAC est qu'il permet de séparer les responsabilités dans un système [19]. Par exemple, un utilisateur peut être assigné à un rôle qui lui donne accès à des données sans lui accorder les permissions de les modifier, ce qui est une garantie essentielle de la sécurité. Contrairement aux modèles comme le DAC (Discretionary Access Control), RBAC permet une gestion plus structurée et sécurisée des accès, car il n'est pas basé sur la discrétion des utilisateurs propriétaires de ressources, mais sur des règles bien définies au niveau organisationnel.

2.2.3.2. Noyau RBAC (RBAC 0)

Le noyau RBAC, ou Core RBAC, ou RBAC0, est le niveau de base du modèle RBAC. Il contient les fonctionnalités essentielles du modèle, dont le concept principal est que les utilisateurs (*U*) se voient assigner des rôles (*R*), et ces rôles sont associés à des permissions (*P*).

Dans ce noyau, les utilisateurs se voient attribuer des rôles via la relation *UA* (User Assignment), et les rôles sont reliés à des permissions via la relation *PA* (Permission Assignment). Ces permissions définissent les actions qu'un utilisateur peut effectuer sur un objet, même si la notion d'objet n'est pas incluse directement dans le noyau RBAC pour garder le modèle général [20].

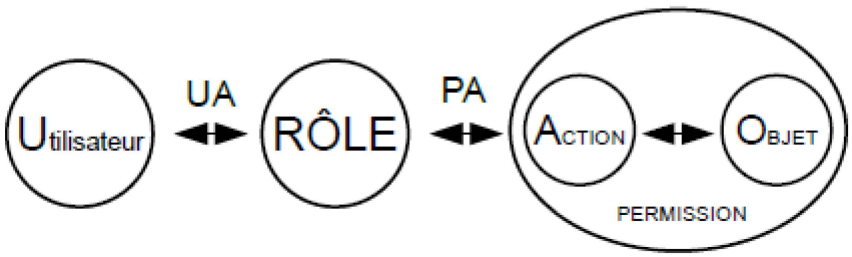


Figure 5 : Noyau RBAC (extrait de [20])

- Utilisateur *(U)* : Représente une entité humaine ou une autre entité qui souhaite interagir avec le système. Dans le modèle RBAC, les utilisateurs ne se voient pas attribuer directement des permissions, mais plutôt des rôles.
- Rôle *(R)* : Un rôle correspond à une fonction spécifique au sein de l'organisation, par exemple "administrateur" ou "employé". Les utilisateurs sont affectés à des rôles en fonction de leurs responsabilités dans l'organisation. Le lien *UA* (User Assignment) entre l'utilisateur et le rôle montre que chaque utilisateur est assigné à un ou plusieurs rôles.
- Permission *(PA)* : Un rôle est lié à une permission via l'assignation de permission *PA* (Permission Assignment). Une permission est composée de deux éléments : l'Action et l'Objet.
- Action (*A*) : Ce sont les opérations qu'un utilisateur peut effectuer, par exemple "lire", "écrire", "modifier".
- Objet (*O*) : Ce sont les ressources sur lesquelles les actions sont exécutées, comme un fichier, une base de données ou une application.

Action sur l'objet : La figure 5 montre qu'une action est liée à un objet via une permission. Cela signifie que chaque rôle permet à un utilisateur de réaliser des actions spécifiques sur des objets particuliers.

Le modèle RBAC permet aussi de définir l'héritage entre rôles et des contraintes sur les combinaisons des rôles. Ces caractéristiques ne seront pas mentionnées dans ce mémoire et pour cette raison elles ne sont pas décrites dans ce chapitre.

2.2.3.3. Conclusion sur le modèle RBAC

Le modèle RBAC a fait l'objet de nombreuses discussions dans la littérature et est, dans nombreuses variantes, amplement utilisé dans la pratique. Il offre une relation cohérente entre la sécurité des données, la structure organisationnelle et les fonctions au sein d'une organisation. Des exemples seront abordés concernant ce modèle qui est une des considérations principales notre projet. Dans le prochain chapitre, nous introduirons une méthode de modélisation des politiques de sécurité qui repose sur ce modèle.

2.2.4. Modèle ABAC

L'ABAC (Attribute-Based Access Control) est un modèle logique de contrôle d'accès qui détermine les autorisations d'accès à des objets en évaluant les règles définies à partir des attributs des sujets, des objets et des conditions environnementales. Les attributs sont des caractéristiques sous la forme de paires nom-valeur qui définissent les entités (ex. : utilisateurs ou ressources) impliquées dans une requête d'accès. L'ABAC se distingue par sa capacité à définir des politiques complexes qui intègrent une diversité d'attributs et de conditions pour exprimer des autorisations adaptées à des contextes spécifiques [21].

Les composants clés du modèle ABAC se présentent inclure comme suit [21] , faisant l'hypothèse que les entités concernées soient des sujets et des objets :

- Attributs du sujet : Les attributs d'un sujet peuvent inclure des propriétés telles que le rôle, le département ou la localisation géographique. Ces informations permettent d'identifier les caractéristiques qui influencent les décisions d'accès.
- Attributs de l'objet : Les objets, tels que des fichiers ou des bases des données, se voient assigner des attributs comme le type de ressource, le niveau de classification ou le propriétaire.
- Conditions environnementales : Ces facteurs peuvent inclure des informations contextuelles telles que l'heure, la location ou le niveau de menace, qui peuvent modifier les décisions d'accès.
- Politiques de contrôle d'accès : Les politiques contiennent les règles qui gouvernent les opérations autorisées, elles sont exprimées comme fonctions booléennes sur des combinaisons d'attributs des sujets, des objets, et de l'environnement.

Dans le fonctionnement du modèle ABAC, lorsqu'un sujet fait une requête d'accès à un objet, le mécanisme de contrôle d'accès ABAC (ACM) exécute les étapes suivantes [21] :

- Récupération des Attributs : Les attributs du sujet, de l'objet, et de l'environnement sont collectés à partir du Policy Information Point ou *PIP*
- Évaluation des Politiques : Les attributs sont utilisés pour évaluer les règles définies dans les politiques pour déterminer si l'accès est autorisé. Les politiques sont administrés par un Policy Administration Point ou *PAP*.

- Décision et Application : La décision est prise par le "*Policy Decision Point" (PDP)* et appliquée par le *Policy Enforcement Point (PEP)*.

Prenons un exemple de scénario pratique : Accès aux dossiers médicaux.

Dans un hôpital, le système ABAC gère l'accès aux dossiers médicaux des patients. Les décisions d'accès sont basées sur les attributs des utilisateurs (sujets), des dossiers (objets) et des conditions environnementales.

Politiques d'accès

Politique 1 (Permis) **:** Les médecins attribués à un service spécifique (ex. : cardiologie) peuvent consulter les dossiers des patients de ce service.

- Condition : L'attribut Rôle du sujet doit être "Médecin".
- Attributs vérifiés :
    - Sujet : Rôle = Médecin, Service = Cardiologie
    - Objet : Type = Dossier Médical, Service = Cardiologie
    - Environnement : Heure dans l'intervalle 08h à 18h.
- *Effet : "Permis".*

2. *Politique 2 (Deny)* : Les employés administratifs ne peuvent pas accéder aux dossiers médicaux.

- Condition : Si l'attribut Rôle du sujet est "Personnel Administratif".
- Attributs vérifiés :
    - Sujet : Rôle = Personnel Administratif

        - Objet : Type = Dossier Médical

- *Effet : « Refuser ».*

Exemple de décision

Cas 1 : Requête acceptée

- Requête : Le docteur Luigi Logrippo demande à lire le dossier d'un patient.
- Attributs :
    - Sujet : Nom = Luigi Logrippo, Rôle = Médecin, Service = Cardiologie.
    - Objet : Type = Dossier Médical, Service = Cardiologie.
    - Environnement : Heure = 10h.
- Évaluation :
    - Politique 1 s'applique. Les conditions sont satisfaites.
    - *Effet : "Permis".*
- Résultat : Le docteur Luigi Logrippo peut avoir accès au dossier.

Cas 2 : Demande refusée

- Demande : Mr Ripama Mohamed, employé administrative, demande à lire un dossier médical.
- Attributs :
    - Sujet : Nom = Ripama Mohamed, Rôle = Personnel Administratif.
    - Objet : Type = Dossier Médical.

- Environnement : Heure = 14h.

- Évaluation :

  - Politique 2 s'applique. Les conditions sont satisfaites.

  - *Effet : « Refuser ».*

- Résultat : Mr Ripama Mohamed ne peut pas accéder au dossier.

  Effet "Deny" : Utilisé pour interdire n accès lorsqu'une combinaison d'attributs satisfait une règle qui conduit à un *Deny*, Ce mécanisme garantit une sécurité stricte en correspondant à tout accès non autorisé.
  Effet "Permit" : Appliqué lorsque toutes les conditions spécifiées par une politique sont respectées, permettant ainsi l'exécution de l'opération demandée.
  Dans le cas où, pour une requête donnée, il n'y a dans les politiques ni une règle conduisant à une permission, ni une règle conduisant à un refus, la décision du *PDP* sera *Not Applicable*. Ce type de décision pourra devenir un refus par effet d'une règle ultérieure comme p.ex. *Deny If not Permit*, mais nous n'irons pas plus loin dans la description des détails de ABAC.
- Ces décisions sont rendues par le *Policy Decision Point (PDP)* et appliquées par le *Policy Enforcement Point (PEP)*, garantissant une gestion des accès précise et dynamique [21].

2.2.4.1. Conclusion sur le modèle ABAC

Le modèle ABAC offre une grande flexibilité et précision en permettant des décisions d'accès basées sur un éventail varié d'attributs, ce qui le rend particulièrement adapté aux scénarios complexes et dynamiques. Il assure également une évolutivité remarquable en permettant des mises à jour dynamiques des politiques par simple ajustement des attributs, sans modification des relations fixes entre les sujets et les objets. De plus, l'ABAC facilite la prise en charge des utilisateurs externes en utilisant des attributs standardisés ou partagés. Cependant, sa mise en

œuvre présente des limites, notamment une complexité opérationnelle nécessitant une gestion rigoureuse des attributs et des politiques, ainsi que des coûts initiaux élevés liés à la modernisation des infrastructures existantes [21].

2.3. CONCLUSION

Ce chapitre a présenté une revue détaillée des principaux modèles de contrôle d'accès, en mettant en évidence les modèles DAC, MAC (incluant Bell-LaPadula), ABAC et RBAC, et en expliquant leur pertinence dans les systèmes complexes comme l'Internet des Objets (IoT). Chaque modèle offre des avantages spécifiques selon le contexte : le DAC permet une gestion flexible mais vulnérable des autorisations, tandis que les modèles MAC, tels que Bell-LaPadula, garantissent la confidentialité dans des systèmes multi-niveaux. Le modèle RBAC, qui est l'un de modèles centraux de notre projet, allie scalabilité, gestion centralisée et efficacité dans des environnements où les rôles et autorisations sont bien définis, tout en réduisant les erreurs de configuration. Enfin, le modèle ABAC se distingue par sa capacité à définir des autorisations dynamiques basées sur des attributs, offrant une flexibilité accrue.

Dans le chapitre suivant, nous aborderons le cadre théorique de notre approche, dans lequel nous décrirons les méthodologies basées sur celle-ci.

CHAPITRE 3 : CADRE THÉORIQUE

Nous utilisons dans ce travail les résultats de Logrippo et Stambouli ainsi que la traduction du documents [22], [23], [24], [25]. Nous resterons très fidèles aux textes originaux.

3.1. INTRODUCTION

La sécurité des données, et en particulier le contrôle des flux d'information, repose sur des principes fondamentaux visant à garantir la confidentialité, l'intégrité et l'accès autorisé. Dans ce cadre, l'utilisation des concepts de théorie des ordres, tels que les préordres et les ordres partiels, permet de formaliser les relations entre entités dans un système. Une telle formalisation simplifie la modélisation des accès et des partages de données, en s'appuyant sur des propriétés telles que la réflexivité et la transitivité des relations.

Le concept de relation *CanFlow (CF)* joue un rôle central dans cette approche. Il définit la fermeture transitive et réflexive d'une relation binaire arbitraire appelée *Channel*, qui représente les canaux d'interaction entre entités. La relation *CF* est un préordre qui peut être analysé pour identifier des classes d'équivalence entre entités, lesquelles sont ensuite ordonnées partiellement. Cette représentation est particulièrement utile dans des systèmes complexes, comme l'Internet des objets (IoT) ou les modèles de contrôle d'accès comme RBAC, car elle permet de déduire efficacement les relations d'accès.

L'utilisation d'algorithmes tels que celui de Tarjan facilite l'identification des classes d'équivalence et leur organisation en ordres partiels. L'efficacité théorique de cet algorithme, qui s'exécute en temps linéaire O(|V|+|E|), c'est à dire linéaire en fonction du nombre d'entités V et canaux E en fait un choix idéal pour traiter les réseaux à grande échelle. Dans ce chapitre, nous décrivons les fondements théoriques généraux ainsi que les spécificités pour les modèles de sécurité tels que RBAC.

3.2. CADRE THEORIQUE GÉNÉRAL

3.2.1. Définition des concepts de base

4. Réseau et relations de canal : Un réseau *N* est défini comme un ensemble fini d'entités avec une relation binaire appelée *Channel*. Chaque entité est identifiée par un nom unique. Soit $Channel \subseteq E \times E$. Cette relation représente les canaux possibles d'interaction entre les entités, comme la lecture ou l'écriture de données.
5. Relation *CF* : est la fermeture réflexive et transitive de *Channel*, ce qui en fait un préordre. $CF = Channel^{+} \cup \{(x, x) / x \in E\}$, où $Channel^{+}$ représente la fermeture transitive de *Channel*
6. Classes d'équivalence : Deux entités sont équivalentes si elles sont connectées mutuellement par *CF*. Une classe d'équivalence contenant des entités *x* et *y* sera notée *[x,y ....]*.

Par exemple deux entités *x* et *y* sont considérées comme *équivalentes* si *CF (x, y)* et *CF (y, x)*. L'ensemble des entités équivalentes forme une *classe d'équivalence*, qui peut être notée *[x, y]*.

7. Un résultat de la théorie des ordres, fondamental dans notre travail, dit qu'un préordre induit un ordre partiel de classes d'équivalences, comme nous verrons dans nos exemples.

    1. Pour chaque classe d'équivalence [x], on associe un label *Lab([x]),* défini comme l'union des labels des classes dominées :
       $Lab([x]) = \cup\, OwnLabel([y])$, où $OwnLabel([y]) = \{Name(z) / z \in [y]\}$.
    2. Pour une entité x, on prend *Lab(x)=Lab([x]).*

Résultats fondamentaux :

Théorème 1 : La relation entre les classes d'équivalence des entités est un ordre partiel. Cela signifie qu'elle est réflexive, transitive et antisymétrique, et elle est notée $\sqsubseteq$. Par ailleurs, $CF(x,y) \Leftrightarrow [x] \sqsubseteq [y] \Leftrightarrow Lab(x) \subseteq Lab(y)$. L'ordre partiel des classes d'équivalence est isomorphe à celui des étiquettes, ordonné par inclusion des sous-ensembles. Les étiquettes propres des classes d'équivalence sont uniques et définissent l'ordre par inclusion : $[x] \sqsubseteq [y] \Leftrightarrow Lab([x]) \subseteq Lab([y])$. Pour une entité *x,* nous prenons *Lab (x) = Lab([x]).*

Théorème 2 : Les relations de canal, les relations *CF*, les ordres partiels et les ensembles d'étiquettes peuvent être interconvertis via des algorithmes linéaires ou polynomiaux, selon le contexte (e.g., fermeture transitive pour *CF*, inclusion pour les étiquettes).

3.2.2. Calcul de relations et structures

Pour l'ensemble ou un ensemble d'étiquettes d'entités dans l'ensemble, ils peuvent être calculés avec des algorithmes en temps linéaire ou en temps polynomial.

Les étapes qui montrent la structure initiale donnée (*Channel*, *CF*, ordre partiel, ou les étiquettes), sont :

1. De la relation *canal* à la relation *CF* :
   À partir de la relation Canal, la relation *CF (CanFlow)* peut être calculée en utilisant des algorithmes de fermeture transitive [26]. Ces algorithmes ont une complexité cubique, qui est polynomiale.

2. De la relation *CF* à l'ordre partiel :
   Les classes d'équivalence (composantes fortement connexes) et leur ordre partiel sont calculés via des algorithmes de composantes fortement connexes, comme celui de Tarjan avec une complexité temporelle linéaire (O(|V|+|E|).

3. De l'ordre Partiel aux étiquettes :
   Étant donné un ordre partiel de classes d'équivalence, les étiquettes des classes d'équivalence et des entités peuvent être calculées en utilisant un algorithme de parcours d'ordre partiel procédant du bas vers le haut en temps linéaire.

4. Des étiquettes à la relation *CF* :
   Étant donné un ensemble d'entités étiquetées, le relation *CF* ou l'ordre partiel des classes d'équivalence peut être calculé en vérifiant l'inclusion parmi les paires d'étiquettes. L'inclusion d'ensemble est un problème ayant la même complexité que le tri, qui est en temps logarithmique [27].

5. De *CF* au *canal*

   Une relation de *canal* peut être calculée à partir d'une relation CF, de manière très simple en définissant le *canal (x, y) = CF (x, y)*. Cela donnera tous les canaux possibles. Les relations de *canal* réduites peuvent ensuite être calculées par des algorithmes de réduction transitifs, de complexité cubique [28], cependant, cela pourrait supprimer des canaux qui pourraient être utiles pour la mise en œuvre.

3.2.3. Exemple avec des concepts des données

Intuitivement, *Channe*l *(x, y)* doit être interprété comme signifiant que les données peuvent se déplacer de l'entité x vers l'entité y. Ainsi, le canal désigne une autorisation, une permission ou un droit et non l'exécution d'une opération. Il peut désigner une permission de contrôle d'accès (une valeur vraie dans une matrice de contrôle d'accès [10] ou une possibilité de lecture ou d'écriture de données par l'utilisation de méthodes de chiffrement-déchiffrement. De nombreuses méthodes existent pour spécifier les relations de *canal*. Lorsque toutes les variables sont fixes, un système DAC, un système RBAC, un système ABAC, une table de routage ou un ensemble d'entités communiquant par chiffrement et déchiffrement définissent tous implicitement des matrices de contrôle d'accès [25]. Ainsi, en termes d'opérations réelles, *Channel (x, y)* peut désigner une autorisation d'écriture de *x* sur *y*, de lecture de *y* depuis *x*, d'envoi de *x* vers *y*, de réception de *y* depuis *x*, ou similaire pour pousser et tirer, placer et obtenir, etc. Une classe d'équivalence est un ensemble d'entités autorisées à partager toutes les données. Les étiquettes identifient les catégories de données auxquelles les entités ont accès, ou la provenance de leurs données.

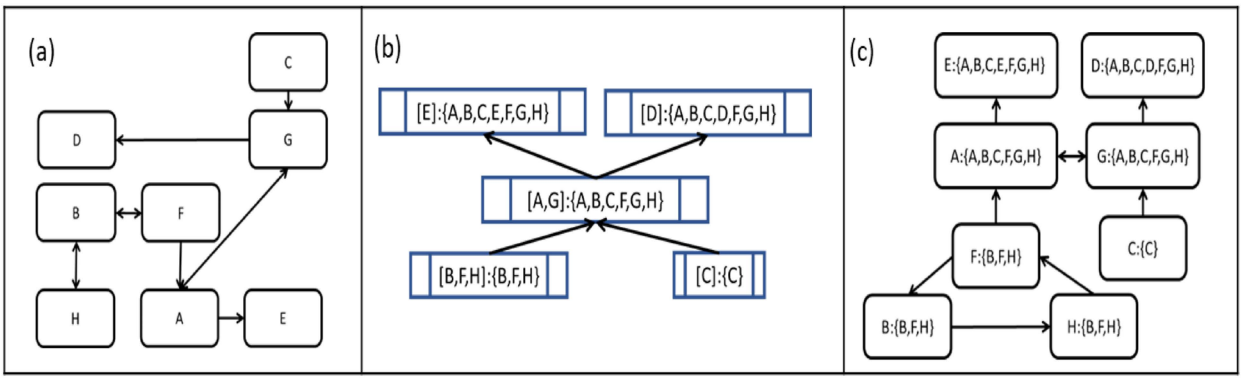


Figure 6: Un *canal* relation (a), ses classes d'équivalence étiquetées (b) et ses entités étiquetées (c).

Dans les diagrammes de la Fig. 6, nous voyons :

- dans (a) un arbitraire *canal* relation entre entités ;
- dans (b) l'ordre partiel correspondant $\sqsubseteq$ des classes d'équivalence étiquetées d'entités qui est une relation entre les classes d'équivalence d'entités, c'est-à-dire une relation réflexive, transitive et antisymétrique, avec des étiquettes (déf. 4) précédées de deux points ;
- dans (c) une attribution d'étiquettes à des entités (Déf. 5), avec des étiquettes précédées de deux points et avec une valeur réduite *canal* relation cohérente avec (b).

Les relations d'ordre sont présentées dans (b) et (c) avec les éléments les plus grands au-dessus des plus petits.

Pour des raisons de lisibilité, ils sont représentés transitivement réduits et sans bords réfléchissants.

Notons les points suivants :

- Les classes d'équivalence non triviales suivantes ont été détectées : [B, F, H] et [A, G].
- D'après Th.1, tous les diagrammes de la Fig. 6 spécifient le même flux de données. En particulier, les libellés dans (b) et (c) définissent le flux de données dans (a).
- Dans la théorie établie, *CF (x, y)* ssi $Lab(x) \subseteq Lab(y)$ caractérise les systèmes multi-niveaux dans un cadre en treillis ; c'est vrai pour tout réseau dans notre théorie.
- Si nous définissons une classe d'équivalence de secret maximal (également appelée secret supérieur ou confidentialité supérieure) d'entités comme étant une classe d'équivalence qui n'a pas de flux de données sortants (c'est-à-dire qu'elle n'est pas strictement dominée par une autre), nous voyons deux classes d'équivalence de secret maximal, [D] et [E]. L'étiquette de ces entités n'est incluse dans aucune autre.
- Si nous définissons une classe d'équivalence d'intégrité maximale d'entités comme étant une classe d'équivalence d'entités qui n'a pas de flux de données entrants (qui ne domine strictement aucune autre) (dans Bell, LaPadula [4], ou dans Sandhu [16]), nous voyons deux classes d'équivalence d'intégrité maximale d'entités, [B, F, H] et [C]. L'étiquette de ces classes d'équivalence est simplement leur propre étiquette.

- Si nous définissons deux entités comme étant en conflit s'il n'existe aucune entité vers laquelle elles peuvent toutes affluer, alors les entités E et D sont en conflit.
- Un ordre partiel de classes d'équivalence d'entités telles que (b) définit une classe d'équivalence de relations de *canal*. (a) et (c) appartiennent tous deux à cette classe d'équivalence, ou peuvent être considérés comme des implémentations de (b). Certains canaux qui ne sont pas représentés dans (a) ou (c) pourraient toutefois être utiles dans la pratique, en fonction des contraintes d'implémentation sur le réseau physique sur lequel la relation *CF* doit être implémentée et d'autres contraintes telles que la vitesse du *canal*.

3.3. CADRE THÉORIQUE SPÉCIFIQUE : RBAC

Dans le cadre du contrôle d'accès, les réseaux bipartites et le modèle RBAC (Role-Based Access Control) jouent un rôle clé en structurant les relations entre les sujets, les objets, et les autorisations, ce qui est particulièrement pertinent dans notre projet.

- Les objets considérés (*o*) sont des entités porteuses de données telles que des fichiers ou des bases de données ;
- Les sujets (*s*) sont des entités capables de lire ou d'écrire des objets ; ils peuvent également véhiculer des données obtenues à partir d'objets par l'effet d'opérations de lecture *r* ou qui leur sont transmises par leurs utilisateurs ;
- Les opérations considérées sont la lecture ***r*** et l'écriture *w* d'objets ;
- Les autorisations (Permissions) permettent aux sujets de lire à partir d'objets (*CanRead* ou *r*) et aux sujets d'écrire dans des objets (*CanWrite* ou *w*) ; ceux-ci définissent une relation *Channel*, voir ci-dessous ;
- Nous nous concentrons sur RBAC 0.
- Nous commençons par une définition générique :

Définition 1 : Un réseau de données bipartite (ou simplement un réseau bipartite désormais) est un réseau d'entités nommées partitionnées en deux sous-ensembles *SUBS s* et *OBS o.* La relation *Channel* est un sous-ensemble de $OBS \times SUBS \cup SUBS \times OBS$. Comme tous les réseaux, les réseaux bipartites peuvent être définis par des matrices de contrôle d'accès.

Définition 2 : On dit qu'une classe d'équivalence *[x]* est *plus secrète* qu'une classe d'équivalence *[y] si [y]* $\subsetneq$ *[x]* ; dans ce cas, *Lab(y)* $\subsetneq$ *Lab(x)* et on dit aussi que *x* est plus secret que *y.* On dit qu'une classe d'équivalence *[x]* a plus d'intégrité qu'une classe d'équivalence *[y] si [x]* $\subsetneq$ *[y]* ; dans ce cas, *Lab(x)* $\subsetneq$ *Lab(y)* et on dit aussi que *x* a plus d'intégrité que *y.* Si *[x]* est un élément maximal (minimal) dans un ordre partiel, on dit que *[x]* ou *x* est de secret maximum *(*intégrité maximale*).*

Par cette définition, la relation *CF* est définie pour les réseaux bipartites et la théorie de la définition 2 s'applique. En particulier, des classes d'équivalence et des étiquettes sont définies pour les sujets et objets dans ces réseaux, selon les définitions ci-dessus.

Nous utiliserons la lettre *s* (resp. *o)* avec des nombres premiers ou des indices pour les variables désignant les membres de *SUBS* (resp. *OBS)*. Les noms d'entités sont des chaînes arbitraires de caractères avec des initiales majuscules. Les noms des sujets seront *S1, S2,... etc,* et pour les objets *O1, O2,.... etc.* De plus, les rôles, voir ci-dessous, auront des noms qui seront souvent écrits *R1, R2, ... etc.* Les variables des rôles seront notées par la lettre *r* avec des nombres premiers et des indices.

Définition 3. Dans un réseau bipartite, on peut définir les deux relations *CR* et *CW* sur *SUBS* $\times$*OBS* comme suit [3]:

a) *CR(s,o) si Canal(o,s)* et

b) *CW(s,o) si Canal(s,o).*

Propriété 1. Dans les réseaux bipartites :

1) *CF(s,s ')* ssi il existe *o* tel que *CF(s,o)* et *CF(o,s ')*
2) *CF(o,o ')* ssi il existe *des s* tels que *CF(o,s)* et *CF(s,o ')*

Preuve. Par le fait qu'il ne peut y avoir de canaux entre sujets ou objets.

Les systèmes RBAC sont des ensembles de définitions RBAC qui peuvent évoluer au fil du temps par une action administrative ou utilisateur. Dans une configuration RBAC, toutes les définitions sont fixes. Les changements dans les définitions seront appelés reconfigurations. Les réseaux RBAC sont définis pour les configurations RBAC. Le réseau RBAC pour une

configuration RBAC est un réseau bipartite qui possède les mêmes ensembles *SUBS* et *OBS* comme configuration RBAC, et une relation *de canal* définie selon les définitions RBAC dans la configuration. Ainsi, une configuration RBAC définit un réseau RBAC et un réseau RBAC définit une configuration RBAC.

Définition 4. Soit *Noms* l'ensemble de tous les noms d'entités d'un réseau. Nous associons à chaque classe d'équivalence[x]un ensemble, appelé *Lab([x])*, qui est un sous-ensemble de Noms. Pour chaque*[x]*, laisser Marque propre *([x]) = {Nom(y) | y∈[x]}*. Pour chaque[x], laisser *Lab([x]) = ∪{*Propre étiquette*([y]) | [y] ⊑ [x]}*.

Définition 5. L'ensemble RBAC *OPS* est défini ici comme *OPS= {CR, CW}.*

Dans RBAC [5], l'ensemble des autorisations pour les sujets est défini à travers plusieurs expressions. Il y avait des raisons à la manière dont ces définitions ont été formulées, mais nous les simplifions ici en disant que dans chaque configuration, il y a une attribution de rôles aux sujets et une attribution de permissions aux rôles, ce qui conduit à une attribution de permissions aux sujets.

Définition 6. Un réseau RBAC pour une configuration RBAC est un réseau bipartite qui possède les mêmes ensembles *SUBS* et *OBS* que la configuration RBAC et où :

1. *Channel (o, s)* ou *CR (s, o)* est vrai dans *R* si *< CR, o > ∈ SP(s)* est vrai dans la configuration RBAC ;
2. *Channel (s, o) ou CW (s, o)* est vrai *ssi < CW, o > ∈SP(s)* est vrai dans la configuration *RBAC ;*
3. Le *canal* est faux pour toutes les autres paires d'entités dans *R.*

3.3.1. Notation graphique, terminologie

On écrit $s\ (r_1, \ldots, r_n)$ si le sujet *s* a, dans la configuration actuelle, des rôles $r_1 \ldots r_n$. Nous combinons cette notation avec la notation d'étiquette pour écrire $s(r_1, \ldots, r_n):\{y_1, \ldots, y_n\}$ pour

faire référence à la fois aux rôles et à l'étiquette de *s* . Nous avons vu comment le label d'un sujet dans un réseau ou une configuration RBAC peut être calculé à partir de ses permissions, donc de ses rôles.

La représentation graphique des exemples (graphiques de configuration*)* est la suivante (dans les exemples, nous aurons des constantes pour les sujets, les objets et les rôles, avec des initiales majuscules).

Comme d'habitude dans les travaux connexes, la fonction *PA* pour une configuration RBAC est représentée par une matrice de contrôle d'accès qui montre les autorisations *CR* ou *CW* pour chaque rôle sur chaque objet, abrégées en simples *R* et *W*. Nous appelons cette table d'autorisation de rôle.

Les sujets sont représentés sous forme d'ovales avec leurs noms, leurs rôles attribués et leurs étiquettes calculées.

Les objets sont représentés sous forme de rectangles avec leurs étiquettes calculées.

Les flèches dirigées représentent la relation *Channel* ou *CF* entre les sujets et les objets, généralement sous une forme transitivement réduite et sans bords réflexifs. La direction d'une flèche est la direction du transfert ou du flux de données autorisé, et les flèches bidirectionnelles signifient la présence de canaux ou de flux dans les deux sens. Nous n'utilisons pas de flèches différentes pour les relations *Channel* et *CF* dérivées de la transitivité puisque nous nous intéressons uniquement à *CF*, car il pourrait être implémenté par différentes combinaisons de canaux.

Les graphiques de configuration seront généralement disposés dans une direction ascendante, pour montrer le flux de données du plus haut niveau d'intégrité au plus haut niveau de secret.

Pour les configurations RBAC, nous pouvons également donner leurs graphiques d'ordre partiel. De tels graphiques sont constitués de rectangles double face contenant chacun une étiquette et l'ensemble des entités équivalentes qui portent cette étiquette ; les flèches représentent l'ordre partiel d'inclusion dans l'ensemble des étiquettes, et donc également le flux de données entre les classes d'équivalence.

De manière informelle, nous disons que les sujets peuvent connaître les données et que les objets peuvent les stocker, où les données sont des fichiers ou des bases de données. Les sujets peuvent connaître les données de leurs utilisateurs (fonction RBAC *SU)* ou les lire à partir d'objets ; ils peuvent également écrire des données qu'ils connaissent sur les objets. Les objets peuvent stocker leurs propres données et celles qu'ils obtiennent grâce à l'effet des sujets qui écrivent dessus. Nous nous intéressons ici aux flux de données potentiels, et donc uniquement à la possibilité de lecture ou d'écriture, plutôt qu'aux opérations de lecture ou d'écriture réellement réalisées.

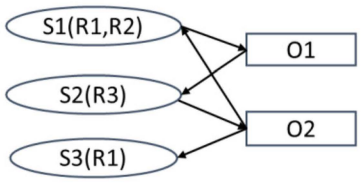
Nous utiliserons les intuitions suivantes :

- *CF(s,o)* implique que toutes les données que *s* peut connaître, *o* peuvent stocker ;
- *CF(o,s)* implique que toutes les données que *o* peut stocker, *s* peuvent le savoir ;
- *CF(s,s')* implique que toute donnée que *s* peut connaître, *s'* peut également la connaître ;
- *CF(o,o')* implique que toutes les données que *o* peut stocker, *o'* peuvent également stocker.

Un premier exemple suit pour présenter la notation et quelques idées de base de méthode que nous utilisons.

La figure 7.a) montre un tableau de rôles et d'autorisations. La figure 7.b) montre, sous forme de graphe biparti, un réseau avec trois sujets, *S1* avec les rôles *R1* et *R2, S2* avec le rôle *R3* et *S3* avec le rôle *R1*. Les autorisations de lecture et d'écriture des sujets sur les objets sont représentées par des flèches.

	O1	O2
R1		R
R2	W	
R3	R	W



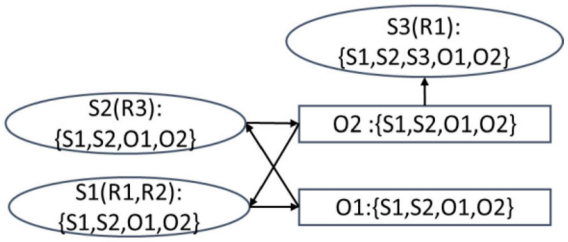


Figure 7. a) Une table de rôles-autorisations, b) son graphe biparti correspondant, et c) son graphe étiqueté correspondant

La Fig. 7.c) montre une représentation de la même configuration mais orientée vers un secret maximum, avec les étiquettes calculées. Il existe deux classes d'équivalence d'entités dans la configuration, c'est-à-dire *[S1, S2, O1, O2] < [S3]*. Aussi *Lab ([S1, S2, O1, O2]) = {S1, S2, O1, O2}* et *Lab([S3]) = {S1, S2, S3, O1, O2}*. Ceci est illustré sur la figure 8.a). Nous pouvons maintenant voir que la configuration décrite jusqu'à présent est équivalente à d'autres, dont l'une est illustrée sur la figure 8.b) pour le tableau rôle-autorisation de la figure 8.c), où seuls deux rôles sont utilisés. L'héritage *R1>R2* peut être défini.

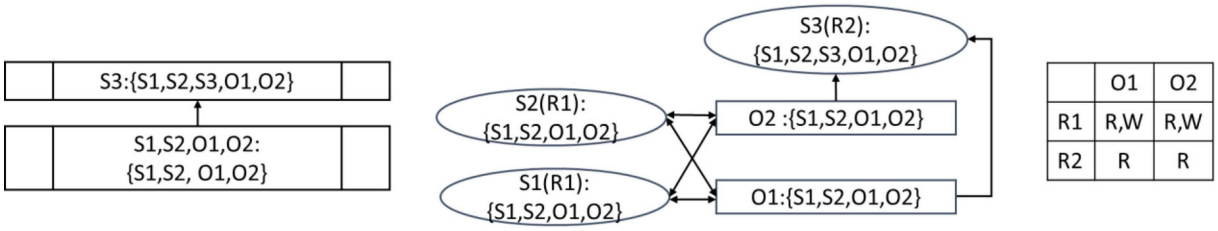


Figure 8. a) Le graphe d'ordre partiel pour le réseau de la Fig.7 ; (b) un réseau équivalent ; c) sa table de rôles-autorisations

Ainsi, un graphe d'ordre partiel est une représentation d'un certain nombre de réseaux et de configurations RBAC équivalents et peut servir de base à une réorganisation de l'ensemble des entités et des rôles. Ceci est important compte tenu des éventuelles contraintes de mise en œuvre, selon lesquelles certaines attributions de rôles ou d'autorisations pourraient être préférables à d'autres.

La figure 7.a) montre un réseau à flux de données réduit, dans le sens où la suppression de toute autorisation modifierait les étiquettes et le flux de données. Dans les réseaux réduits, les données peuvent devoir passer par des entités intermédiaires pour parvenir à des entités capables de les connaître ou de les stocker. La figure 8.b) montre à la place le réseau de flux de données *maximal* dans sa classe d'équivalence, puisque tous les canaux possibles sont présents. Dans les réseaux maximaux, comme celui de la figure 8.b), des autorisations directes sont accordées aux sujets de transférer des données là où ils peuvent aboutir et d'où qu'ils puissent venir. Dans les implémentations, les autorisations accordées dépendent de la structure organisationnelle et des besoins. Nous afficherons généralement des graphiques réduits, afin de réduire l'encombrement des flèches pour plus de lisibilité. Notez que, même si un réseau à flux de données *maximal* est unique dans sa classe d'équivalence, les réseaux réduits ne le sont généralement pas.

Des exemples comme celui-ci, où nous avons un ensemble d'étiquettes totalement ordonné, peuvent être considérés comme des implémentations de systèmes simples Bell-La Padula. Le sujet *S3* pourrait être considéré comme étant au niveau *Secret* puisque ses données ne peuvent être connues nulle part ailleurs, alors que toutes les entités restantes pourraient être considérées comme étant à un niveau inférieur, que l'on peut appeler *Public,* puisque leurs données sont connues partout. Pour l'intégrité, l'inverse est vrai : les entités de la classe d'équivalence de niveau inférieur, qui n'obtiennent pas de données d'autres entités, ont collectivement l'intégrité la plus élevée, tandis que *S3,* qui obtient les données de l'ancienne classe, a l'intégrité la plus faible.

3.4. CONCLUSION

Dans ce chapitre nous avons les descriptions bien détailler des concepts théoriques de notre approche, Ces concepts bien établis de sécurité des données s'appuient directement sur les principes de la théorie des ordres partiels. Ils se déclinent en plusieurs variantes au sein de la théorie de la sécurité des données. L'une de ces variantes repose sur une approche de sécurité des données basée sur les réseaux, où les flux sécurisés doivent respecter une structure en réseau définie par l'inclusion des ensembles d'étiquettes prédéterminées. Toutefois, la théorie développée ici s'applique à tout type de réseau, sans restriction spécifique dont nous nous baserons pour la réalisation de notre système.

Dans le chapitre suivant, intitulé 'Conception du système', nous aborderons la conception de notre système que nous avons développé.

CHAPITRE 4 : CONCEPTION DU SYTÈME

4.1. INRODUCTION

Ce chapitre présente l'architecture et la conception du système développé dans le cadre de ce projet. L'objectif est de montrer comment les différents composants interagissent pour permettre la configuration et le contrôle sécurisé des flux d'information, en intégrant les modèles DAC, MAC, RBAC et ABAC. La conception repose à la fois sur des structures de données efficaces et sur des interfaces interactives permettant aux utilisateurs de tester divers scénarios de sécurité.

4.2. SPÉCIFICATION DES BESOINS

Cette section décrit les besoins du système développé, en les divisant en deux catégories principales : les besoins fonctionnels, qui définissent les services attendus du système, et les besoins non fonctionnels, qui définissent les contraintes de qualité, de performance et d'usage.

4.2.1. Besoins fonctionnels

Les besoins fonctionnels définissent ce que le système **doit faire** pour répondre aux attentes des utilisateurs. Le système doit :

1. **Importer des configurations via un fichier Excel**

    - Charger automatiquement des entités (sujets, objets), permissions, rôles et attributs à partir d'un fichier conforme.

2. **Permettre la saisie de commandes dynamiques**

- Interpréter et exécuter des commandes utilisateur via une interface de type terminal.

3. **Créer, modifier et supprimer des entités**

   - Ajouter ou retirer des sujets, objets, rôles et attributs.

4. **Attribuer ou révoquer des permissions d'accès**

   - Affecter des permissions de lecture (CanRead) et d'écriture (CanWrite) entre sujets et objets selon le modèle DAC.

5. **Appliquer les modèles de sécurité** :

   - **DAC** : Permissions définies par le propriétaire.
   - **MAC** : Étiquettes de sécurité, niveaux de confidentialité.
   - **RBAC** : Attribution de rôles aux sujets.
   - **ABAC** : Définition de règles basées sur des attributs.

6. **Visualiser les graphes de relations**

   - Générer des représentations graphiques des canaux d'accès, héritages de rôles et violations de politiques.

7. **Propager les étiquettes et détecter les violations**

   - Utiliser un algorithme (Tarjan) pour identifier les composantes, propager les étiquettes et vérifier les violations de type "Muraille de Chine".

8. **Afficher les étiquettes propagées dans un tableau**

   - Fournir un résumé des entités avec leurs étiquettes de sécurité calculées.

4.2.2. Besoins non fonctionnels

Les besoins non fonctionnels décrivent les **qualités** du système en matière de performance, fiabilité, ergonomie, etc. Le système doit :

1. **Être interactif et réactif**

    - Répondre aux commandes utilisateur (moins de 1 seconde pour 10 000 entités).

2. **Gérer de grands volumes de données**

    - Supporter plus de 100 000 entités avec une complexité algorithmique maîtrisée

      (rappel : la complexité théorique est de (O (|V| + |E|)), où V est le nombre d'entités et E est le nombre de canaux [25]

3. **Être modulable**

    - Permettre l'ajout futur d'autres modèles ou de nouvelles règles de sécurité sans modifier l'architecture de base.

4. **Offrir une visualisation intuitive**

    - Les graphes doivent être clairs, lisibles, et les permissions non redondantes doivent être bien représentées.

5. **Respecter les contraintes de sécurité**

    - Aucun canal ou permission ne doit violer les règles imposées par les modèles de sécurité en place.

6. **Être robuste aux erreurs utilisateur**

    - Gérer les erreurs de syntaxe ou de logique dans les commandes sans plantage.

4.2.3. Évaluation des contraintes non fonctionnelles

CATÉGORIE	CONTRAINTE	ÉVALUATION
PERFORMANCE	Simulation avec l'algorithme de Tarjan sur différents jeux de données (jusqu'à 200 000 entités).	Assurer un temps de traitement rapide même pour un grand nombre d'entités.
SÉCURITÉ	Aucune permission implicite ; la politique de flux respecte les contraintes DAC, MAC, RBAC et China-Wall.	Conforme aux modèles théoriques.
FIABILITÉ	Le système vérifie chaque commande et signale les erreurs (syntaxe, rôle inexistant, etc.).	Testé sur plusieurs scénarios d'erreurs.
PORTABILITÉ	Application exécutable sur Windows, MacOs et Linux (via navigateur).	Confirmé grâce à Streamlit.
ERGONOMIE	Interface interactive, visualisation immédiate des relations et graphes.	Favorise la compréhension conceptuelle.
MAINTENABILITÉ	Code structuré en modules et classes indépendantes.	Facilite les mises à jour futures.

Tableau 3 : Évaluation des contraintes non fonctionnelles

4.3. CONCEPTION DU SYSÈME

Cette section présente l'architecture interne du système ainsi que sa modélisation à l'aide de diagrammes UML. Ces diagrammes présentent la structure du système, les interactions entre les composants, les flux de données ainsi que les comportements attendus lors de l'exécution des fonctionnalités principales.

La conception repose sur une architecture modulaire, orientée autour de l'application des modèles de contrôle d'accès (DAC, MAC, RBAC, ABAC), d'une gestion dynamique des permissions, et d'une interface interactive.

4.3.1. Architecture globale du système

Le système repose sur trois couches principales :

- **La couche d'entrée utilisateur**, qui permet soit l'importation d'un fichier Excel décrivant les permissions, soit la saisie de commandes dynamiques dans un terminal interactif.

- **La couche de traitement**, qui applique les modèles de sécurité, exécute l'algorithme de Tarjan, propage les étiquettes et détecte les violations de règles (Muraille de Chine).

- **La couche de visualisation**, qui affiche les graphes des entités et des permissions ainsi qu'un tableau résumant les étiquettes de sécurité propagées.

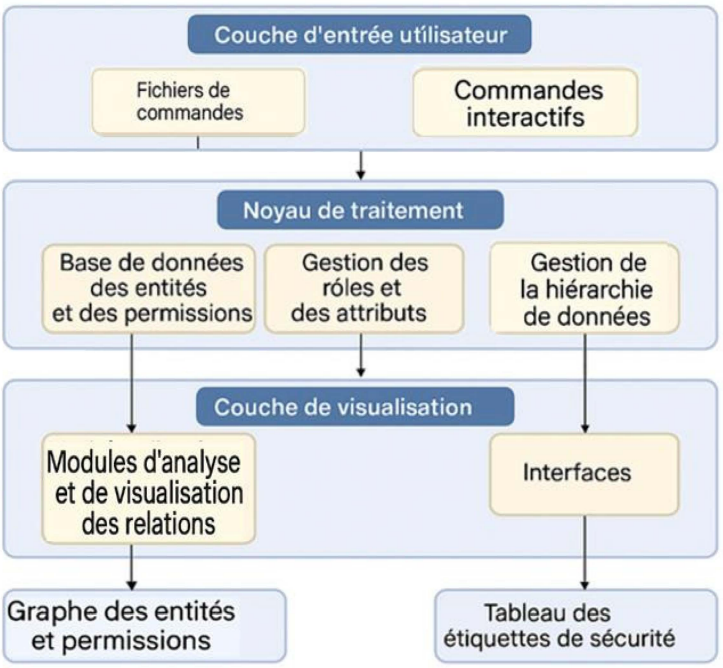


Figure 9: Architecture globale du système

4.1.1. Architecture logicielle (patron MVC)

Le système suit une approche inspirée du patron de conception MVC (Modèle–Vue–Contrôleur) :

- Modèle (Model) : gère les entités de sécurité (sujets, objets, rôles, permissions) et les algorithmes (Tarjan, propagation).
- Vue (View) : représente la partie visuelle (graphiques PyVis, tableaux Streamlit, visualisation des SCC).
- Contrôleur (Controller) : interprète les commandes saisies dans le terminal (apply_prompt()) et met à jour dynamiquement les données et les vues.

Cette séparation permet :

- une extensibilité (ajout futur du modèle ABAC sans perturber les autres) ;
- une réutilisabilité du code (modules indépendants) ;
- une meilleure maintenance du système.

4.1.2. Description des composantes principales

L'architecture générale de notre système repose sur les éléments suivants :

1. Noyau de gestion des modèles de contrôle d'accès : Ce noyau central regroupe les modèles DAC, MAC, RBAC et ABAC. Chaque modèle est géré par un module dédié qui assure l'application des règles de sécurité spécifiques à chaque contexte.

2. Base de données des entités et des permissions : Une base de données relationnelle est utilisée pour stocker les informations sur les sujets, objets, permissions, rôles et attributs. Cette base de données centralise les informations afin de faciliter la gestion et l'application des différentes politiques d'accès. Dans la base de données, nous utilisons le modèle de matrice de contrôle d'accès où les permissions (*CanRead R, CanWrite W*) sont associées directement à des entités spécifiques (sujets et objets).

3. Modules d'analyse et de visualisation des relations : Les outils NetworkX et PyVis sont utilisés pour créer des représentations graphiques claires des canaux et flux de données entre les entités, facilitant ainsi la détection des erreurs de configuration, des redondances ou des incohérences.

4. Module de gestion des rôles et des attributs : Ce module permet de gérer les rôles et les attributs des utilisateurs dans le contexte des modèles RBAC et ABAC que nous utiliserons. Il intègre également des fonctionnalités permettant de créer, supprimer ou modifier les rôles et les attributs, garantissant ainsi une gestion dynamique des permissions.

5. Système de gestion de la hiérarchie des niveaux de sécurité : Ce composant est spécialement conçu pour gérer les niveaux de sécurité et les étiquettes des entités, assurant un contrôle strict des accès selon les niveaux de sécurité établis.

6. Interfaces utilisateur : l'interface utilisateur est composée de deux éléments principaux : l'option de gestion via des fichiers Excel et l'option de commandes manuelles. Cela permet à l'utilisateur de choisir l'option la plus appropriée pour gérer les entités et les autorisations.

4.1.3. Interaction entre les modules

Les modules du système interagissent principalement via des API et des interfaces de commande. Cette architecture modulaire garantit une communication fluide entre le front-end et le back-end tout en facilitant la maintenance et les évolutions futures du système.

1. La gestion des permissions : les permissions sont configurées et stockées dans une matrice d'accès. Lorsqu'une entité souhaite interagir avec un objet, le module vérifie dans cette matrice si la permission est accordée. Les relations entre les entités sont extraites de la matrice et intégrées dans un graphe visuel pour identifier les flux d’information permis ou interdits.

2. Gestion centralisée des autorisations : les modules de gestion des rôles et des autorisations communiquent avec la base de données pour mettre à jour les informations sur les autorisations. Lorsque des utilisateurs activent des rôles lors de leurs sessions, les autorisations correspondantes sont automatiquement attribuées aux sujets concernés.

3. Contrôle d'accès multi-niveaux (MAC) : le module de gestion des niveaux de sécurité interagit avec le noyau MAC pour attribuer des étiquettes de sécurité aux entités et s'assurer que les règles de confidentialité et d'intégrité définies par notre modèle sont respectées. Les interactions entre ces modules sont cruciales pour garantir la mise en œuvre correcte des règles de sécurité dans les environnements multi-niveaux.

4. Visualisation des relations entités-permissions : les modules d'analyse et de visualisation interagissent avec les autres composants pour obtenir les informations nécessaires et générer des représentations graphiques. Cela permet aux administrateurs de surveiller et d'ajuster les autorisations de manière proactive.

4.1.4. Présentation des outils et bibliothèques de programmation

Le système a été développé en Python 3.11, un langage orienté objet favorisant la lisibilité et l'intégration de bibliothèques scientifiques.
Les principales bibliothèques utilisées sont :

- **Streamlit** : pour la création de l'interface graphique interactive. Elle permet la visualisation simultanée des graphes, des tableaux et d'un terminal de commandes intégré.

- **Pandas** : pour la gestion et le traitement des données sous forme de tableaux (DataFrame). Elle facilite la lecture des fichiers Excel, la normalisation des entités et la manipulation des permissions.

- **NetworkX** : pour la modélisation des graphes représentant les flux de données et les relations entre sujets, objets et rôles.

- **PyVis** : pour la visualisation dynamique des graphes avec la possibilité d'interaction (déplacement, zoom, etc.).

- **Matplotlib** : pour la génération des graphiques d'évaluation des performances (comparaison de la complexité temporelle entre Tarjan et la propagation).

- **OpenPyXL** : pour la lecture et l'écriture des fichiers Excel (.xlsx).

L'ensemble du code repose sur une architecture modulaire afin de séparer les responsabilités entre :

- la gestion des données (lecture, normalisation, propagation des rôles),

- la logique métier (implémentation des modèles DAC, MAC, RBAC, China-Wall),

- et la présentation graphique (affichage interactif via Streamlit et PyVis).

- Le système proposé lit et interprète les données d'entrée sous la forme de fichiers de commandes (.xlsx), plutôt que CSV. Ce choix s'explique par la complexité des relations manipulées (sujets, objets, rôles, héritages, etc.) qui nécessitent des colonnes multiples et parfois des feuilles distinctes. Le format de fichier de commandes assure une meilleure lisibilité, une compatibilité native avec les bibliothèques Python utilisées (pandas, openpyxl), ainsi qu'une interopérabilité avec les utilisateurs finaux, qui peuvent préparer et valider leurs données directement dans un tableur avant l'importation.

4.1.5. Identification des acteurs

L'Utilisateur est l'acteur principal du système.
Il représente l'administrateur ou l'opérateur en charge de la gestion des permissions et des configurations de sécurité.

Ses responsabilités comprennent :

- L'importation des fichiers Excel contenant la configuration des entités et permissions.
- La saisie manuelle de commandes dynamiques via le terminal interactif.
- La création, la modification et la suppression des rôles, permissions et attributs.
- Le déclenchement de la génération des graphes et des tableaux de visualisation.
- La surveillance des violations de politiques de sécurité.

4.1.6. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présenté ci-dessus modélise les interactions principales entre l'utilisateur du système et les différentes fonctionnalités offertes par l'outil de gestion des permissions et des flux d'information. Il permet de comprendre, de manière synthétique, le rôle de l'utilisateur ainsi que les opérations essentielles accessibles dans le système.

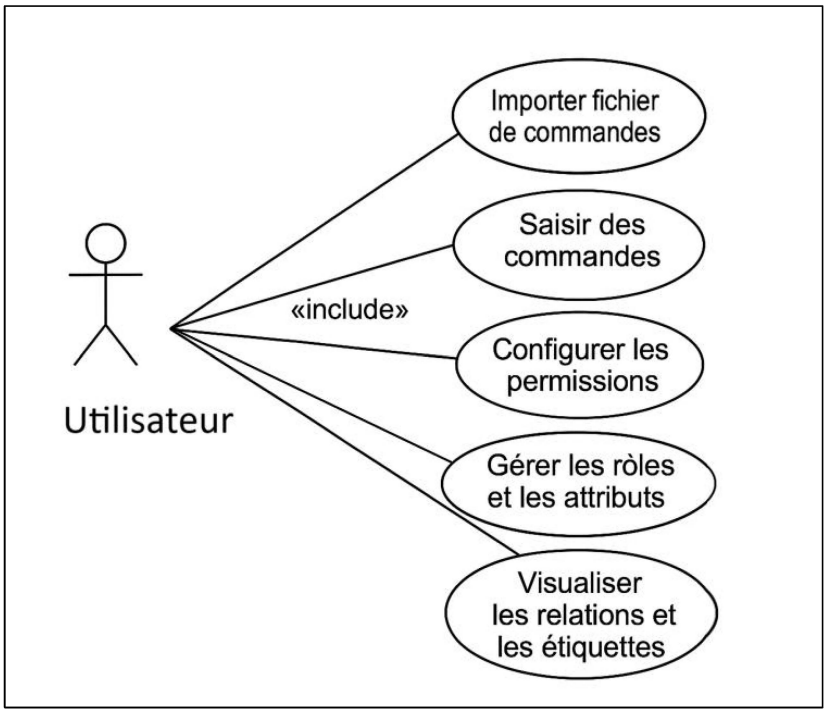


Figure 10: Diagramme de cas d'utilisation

4.1.7. Les Descriptions textuelles du cas d'utilisation

Le tableau suivant est la représentation des différents cas d'utilisation, chaque tableau détaille un cas d'utilisation avec ses composantes classiques : nom, acteur, description, préconditions, scénarios, etc.

Élément	***Description***
Nom	Importer un fichier Excel
Acteur principal	Utilisateur
But	Charger un fichier contenant les entités, permissions et rôles
Préconditions	Le fichier doit être au format .xlsx et respecter la structure attendue
Postconditions	Les entités et permissions sont enregistrées dans la base de données
Scénario principal	1. L'utilisateur sélectionne un fichier Excel 2. Le système lit les données 3. Les entités sont créées dans le système
Exceptions	Format incorrect du fichier, colonnes manquantes, données invalides

Tableau 4: Cas d'utilisation 1 : Importer un fichier Excel

Élément	*Description*
Nom	Saisir des commandes
Acteur principal	Utilisateur
But	Permettre la configuration dynamique via un terminal interactif
Préconditions	Le système doit être prêt à recevoir des commandes
Postconditions	Les entités et permissions sont mises à jour dans le système
Scénario principal	1. L'utilisateur saisit une commande (ex. : AddSub S1) 2. Le système interprète 3. Mise à jour de la base
Exceptions	Commande inconnue, syntaxe invalide, entité inexistante

Tableau 5: Cas d'utilisation 2 : Saisir des commandes

Élément	*Description*
Nom	Configurer les permissions
Acteur principal	Utilisateur
But	Gérer les droits d'accès (lecture, écriture) entre entités
Préconditions	Les entités (sujets et objets) doivent exister
Postconditions	Les permissions sont mises à jour dans la matrice d'accès

Scénario principal	1. L'utilisateur ajoute ou modifie une permission 2. Le système vérifie les règles 3. La permission est enregistrée
Exceptions	Violation de règles (ex : Muraille de Chine), objet ou sujet inconnu

Tableau 6: Cas d'utilisation 3 : Configurer les permissions

Élément	***Description***
Nom	Gérer les rôles et les attributs
Acteur principal	Utilisateur
But	Créer, modifier, ou supprimer des rôles (RBAC) et des attributs (ABAC)
Préconditions	Le système est initialisé avec les entités concernées
Postconditions	Les rôles ou attributs sont mis à jour pour les entités
Scénario principal	1. L'utilisateur crée/modifie un rôle ou attribut 2. Le système valide 3. Mise à jour de la base
Exceptions	Rôle déjà existant, attribut invalide, incohérence de modèle

Tableau 7:Cas d'utilisation 4 : Gérer les rôles et les attributs

Élément	*Description*
Nom	Visualiser les relations et les étiquettes
Acteur principal	Utilisateur
But	Permettre à l'utilisateur de générer et consulter les graphes des relations et les étiquettes propagées
Description	Ce cas d'utilisation permet de visualiser : - Les graphes des entités et des permissions- Les étiquettes de sécurité calculées par l'algorithme- Les violations des règles de sécurité éventuelles
Préconditions	Les entités et permissions doivent avoir été configurées et les modèles appliqués
Postconditions	Les graphes et tableaux sont générés et affichés à l'utilisateur
Scénario principal	1. L'utilisateur déclenche la commande de visualisation.2. Le système récupère les informations actuelles.3. Les graphes sont générés (avec NetworkX et PyVis).4. Les étiquettes sont affichées dans un tableau synthétique.5. Les violations éventuelles sont signalées.
Exceptions	- Aucune donnée configurée (message d'erreur).- Erreur lors de la génération des graphes.

Tableau 8: Visualiser les relations et les étiquettes

4.1.8. Diagramme de séquence

Le diagramme de séquence présenté ci-dessous modélise l'enchaînement des interactions entre les différentes composantes du système lors du processus de configuration et de vérification des permissions d'accès. Il met en évidence l'ordre chronologique des messages échangés entre les entités, illustrant ainsi le comportement dynamique du système.

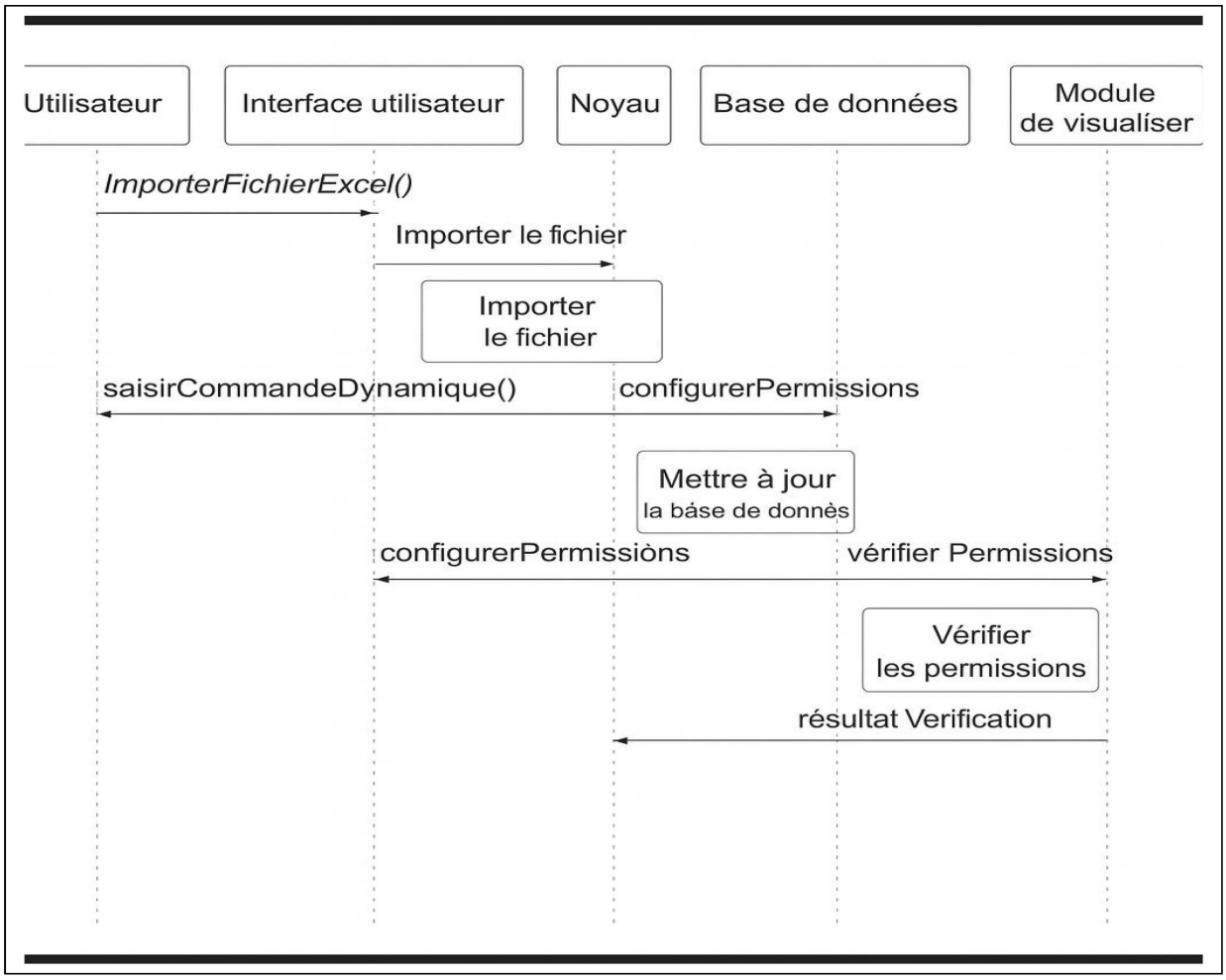


Figure 11: Diagramme de séquence

Élément	*Rôle dans le système*
Utilisateur	Initiateur de l'action. Peut importer un fichier Excel ou saisir des commandes.
Interface utilisateur	Interface graphique ou en ligne de commande permettant l'interaction directe.
Noyau	Partie centrale du système, chargée d'analyser et diriger les commandes.
Base de données	Stocke les entités, permissions, rôles et configurations.
Module de configuration	Applique les modifications demandées par l'utilisateur (attribution de permissions, etc.).
Module de visualisation	Vérifie et affiche les résultats sous forme de graphes ou de tableaux.

Tableau 9:Acteurs et objets impliqués

4.1.9. Diagramme de classe

Le diagramme de classes présenté ci-dessus formalise la structure logique du système de contrôle d'accès développé dans le cadre de ce projet. Il illustre les différentes entités logicielles du système, leurs attributs, méthodes, ainsi que les relations et cardinalités entre elles. Le modèle repose sur une architecture modulaire permettant l'intégration des modèles DAC, MAC, RBAC et ABAC.

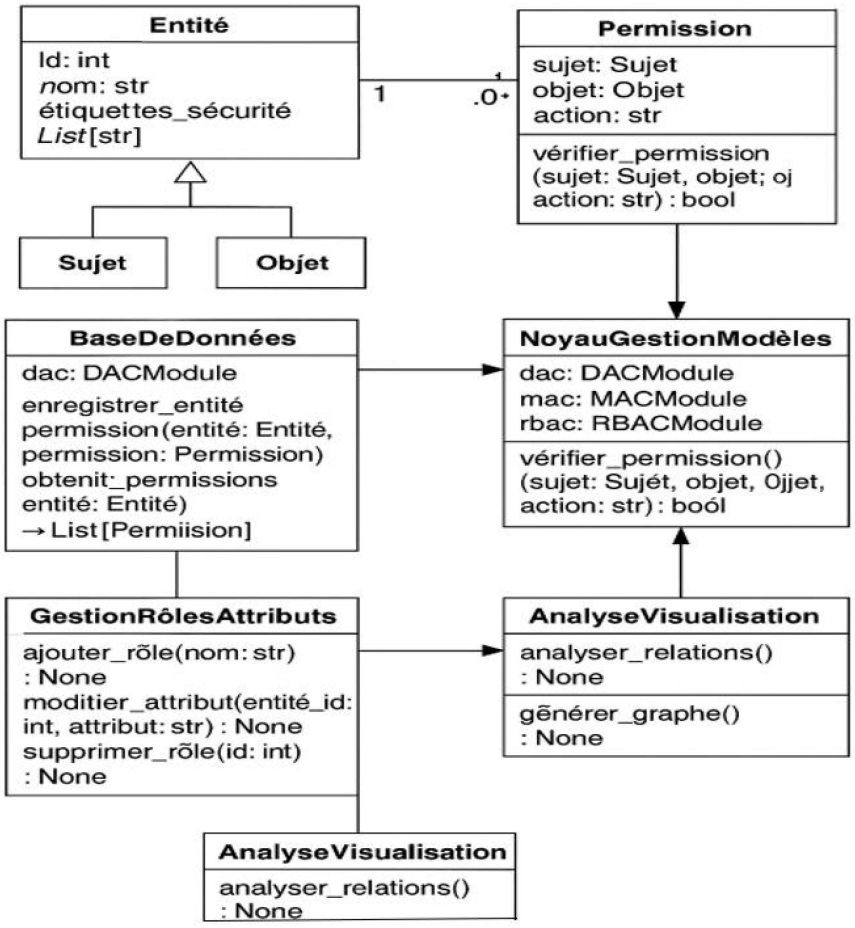


Figure 12: Diagramme de classe

- **Relations et cardinalités**

Le diagramme de classes présenté met en évidence les relations entre les principales composantes du système ainsi que les cardinalités associées :

- **Relation entre Entité (qui peut-être aussi un sujet ou un objet) et Permission :**

  Chaque instance de la classe **Entité** peut être associée à **plusieurs permissions** qui définissent les droits d'accès concernant cette entité.

  La cardinalité est la suivante :

  - Côté Entité : **1** (chaque permission cible exactement une entité unique)

- Côté Permission : **0..*** (une entité peut ne pas avoir de permissions, ou en avoir plusieurs)

- **Association entre BaseDeDonnées et NoyauGestionModèles**
  La classe BaseDeDonnées est liée au NoyauGestionModèles, qu'elle alimente en informations sur les entités et leurs permissions.
  Cette relation matérialise la **dépendance fonctionnelle** : le noyau interroge la base pour vérifier les droits d'accès.
- **Association entre GestionRôlesAttributs et BaseDeDonnées**
  La classe GestionRôlesAttributs dépend de BaseDeDonnées pour enregistrer et mettre à jour les rôles et attributs des entités.
  Cette relation représente l'interface de gestion dynamique des politiques de sécurité.
- **Relation entre AnalyseVisualisation et NoyauGestionModèles**
  Le module AnalyseVisualisation interagit avec le NoyauGestionModèles pour collecter les informations nécessaires à la génération des graphes et à l'analyse des flux d'accès.
  Cette relation illustre la chaîne de traitement qui conduit de la configuration des permissions à leur représentation graphique.
- **Association entre GestionRôlesAttributs et AnalyseVisualisation**
  Cette relation permet de relier les opérations de gestion des rôles et attributs avec leur visualisation, assurant la cohérence entre les configurations et leur affichage.

Ces relations et cardinalités traduisent la **modularité et la cohérence du système**, permettant à chaque composant d'assurer un rôle spécifique tout en interagissant avec les autres. Elles garantissent le respect des règles de sécurité et la centralisation des informations nécessaires au contrôle des permissions.

4.2. LANGAGE DE PROGRAMATION UTILISÉ

Le langage Python a été choisi en raison de sa flexibilité, de sa richesse en bibliothèques et de ses capacités de manipulation de données. Python offre un écosystème robuste pour le

développement de systèmes de gestion des accès avec des bibliothèques telles que Pandas pour la gestion des fichiers Excel, permettant ainsi une extraction et une manipulation efficaces des données des entités, des autorisations et des relations. Il possède également une bonne compatibilité avec des outils de visualisation graphique.

Python offre dans sa bibliothèque l'algorithme de Tarjan de détection de composantes connexes et nous verrons dans la Section de chapitre 5 que l'exécution de ce programme est efficace.

4.3. CONCLUSION

La conception du système présentée dans ce chapitre a permis de structurer de manière claire et détaillée l'architecture, les interactions et les fonctionnalités principales de la plateforme de gestion des permissions et de contrôle des flux d'information.

Grâce à une modélisation UML complète (diagrammes de cas d'utilisation, de classes, de séquence et d'architecture), il est possible de visualiser et de comprendre le fonctionnement global du système, ainsi que les relations entre ses différents modules.

Cette conception met en évidence :

- La modularité du système, qui intègre les modèles DAC, MAC, RBAC et ABAC.
- La capacité à traiter des configurations complexes tout en offrant une interface utilisateur souple et interactive.
- L'importance de la **visualisation** des relations et des permissions, essentielle pour la transparence et la supervision des accès.

Elle constitue une base solide pour la phase de réalisation et d'implémentation, en garantissant que les besoins fonctionnels et non fonctionnels identifiés soient pris en compte de manière cohérente et structurée.

CHAPITRE 5 : REALISATION DU SYTÈME

5.1. INTRODUCTION

La réalisation du système dans le cadre de ce projet se concentre sur l'implémentation d'un système graphique interactif de gestion des canaux, flux et permissions entre entités en nous basant sur le cadre théorique décrit dans le chapitre 3. Le système prend en charge plusieurs modèles de contrôle d'accès (DAC, MAC, RBAC et ABAC en partie), et offre deux principales options d'interaction : l'option fichier Excel (dans le cas de MAC et RBAC) et l'option commandes manuelles. Ces options permettent aux utilisateurs d'ajuster dynamiquement les autorisations et les relations, tout en offrant une visualisation graphique claire et précise de la structure courante du réseau.

Ce chapitre détaille l'implémentation technique, la structure du système et présente des exemples concrets tirés de son utilisation.

5.1.1. Système d'entités et canaux

Dans le cas de système d'entités et canaux dans notre système, une entité est un élément abstrait identifié par un nom unique (par exemple *E1*, *E2*, *E3*, etc.), sans distinction explicite entre sujet et objet.

Chaque entité peut à la fois émettre et recevoir des données à travers des canaux, ce qui permet de modéliser de manière générique les flux d'information dans un graphe orienté. Nous avons utilisé l'option commande pour traiter ce système. Les commandes disponibles sont les suivantes :

Commande	Exemple	Description
AddEnt	*AddEnt E1*	Créer l'entité *E1*

RemoveEnt	*RemoveEnt E1*	Retirer l'entité *E1*.
AddCh	*AddCh E1 E2*	Créer un canal entre *E1* et *E2*
RemoveCh	*RemoveCh E1 E2*	Retirer le canal entre *E1* et *E2*

Tableau 10: Commandes d'exécution (Entité)

5.1.1.1. Création et retrait d'entités

La gestion des entités constitue une composante essentielle du système. Elle permet d'ajouter et de retirer des entités qui participent au réseau d'interactions.

La commande *AddEnt* permet de créer une entité isolée, sans relation initiale avec d'autres éléments. Dans cet exemple, nous allons créer une entité nommée *E1*.

Figure 13: Création Entité

Nous retirons ensuite cette entité *E1* que nous avons créé comme montré dans la Figure 14 :

Figure 14: Retrait d'entité

5.1.1.2. Création et retrait de canaux

Les canaux représentent les relations ou autorisations entre entités. Leur gestion permet de modéliser et d'ajuster dynamiquement les flux d'information.

La commande *AddCh* établit un canal (par défaut une relation de lecture) entre deux entités existantes. Dans cet exemple montré dans la Figure 12, nous allons créer deux entités *E1* et *E2* et établir un canal de *E1* a *E2*, cela veut dire que *E2* peut lire les données de *E1*.

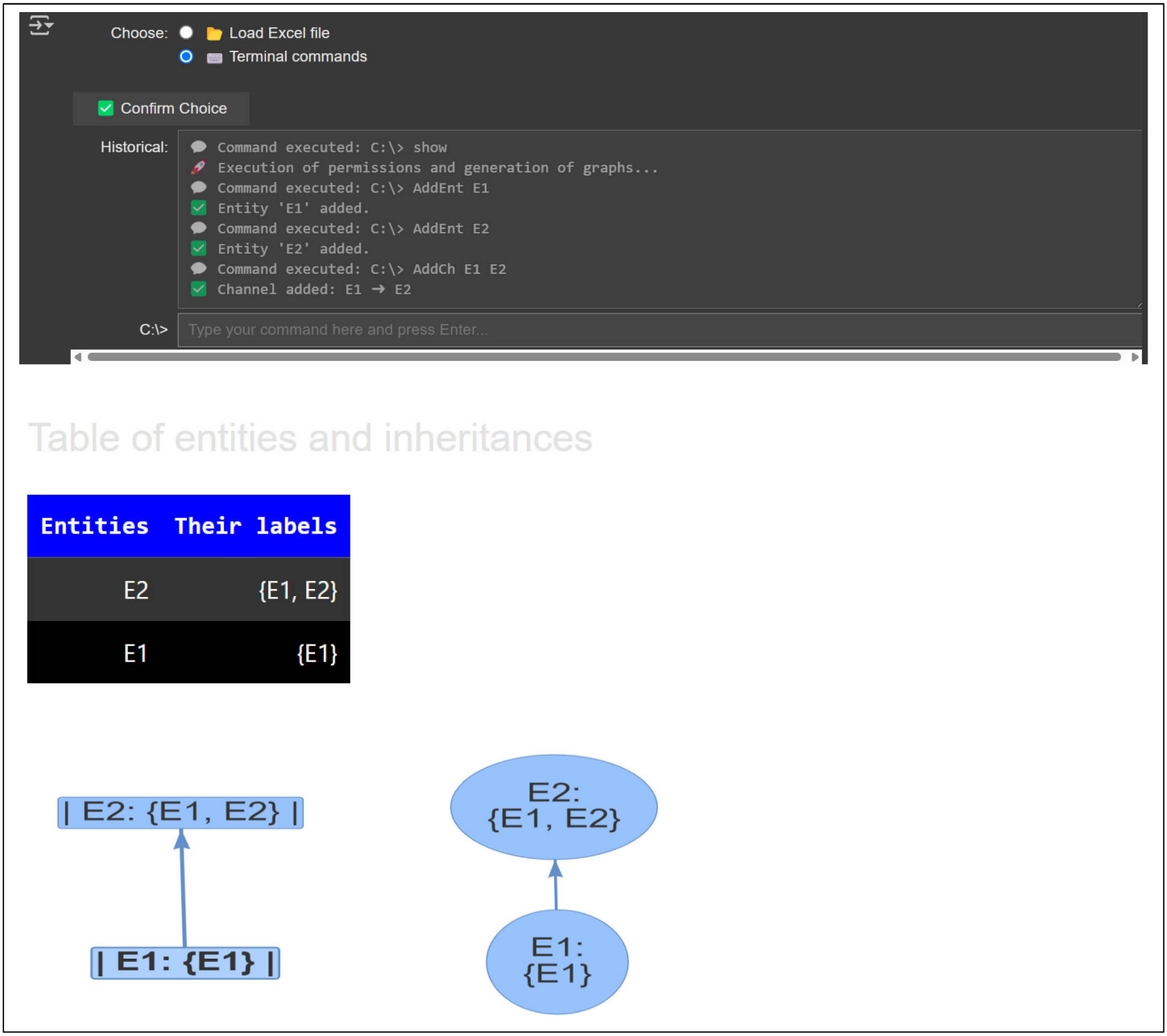


Figure 15: Création canal entre *E1* et *E2*

La commande *RemoveCh* retire uniquement la relation spécifiée entre deux entités, sans retirer les entités elles-mêmes.

Maintenant nous allons retirer le canal entre *E1* et *E2*.

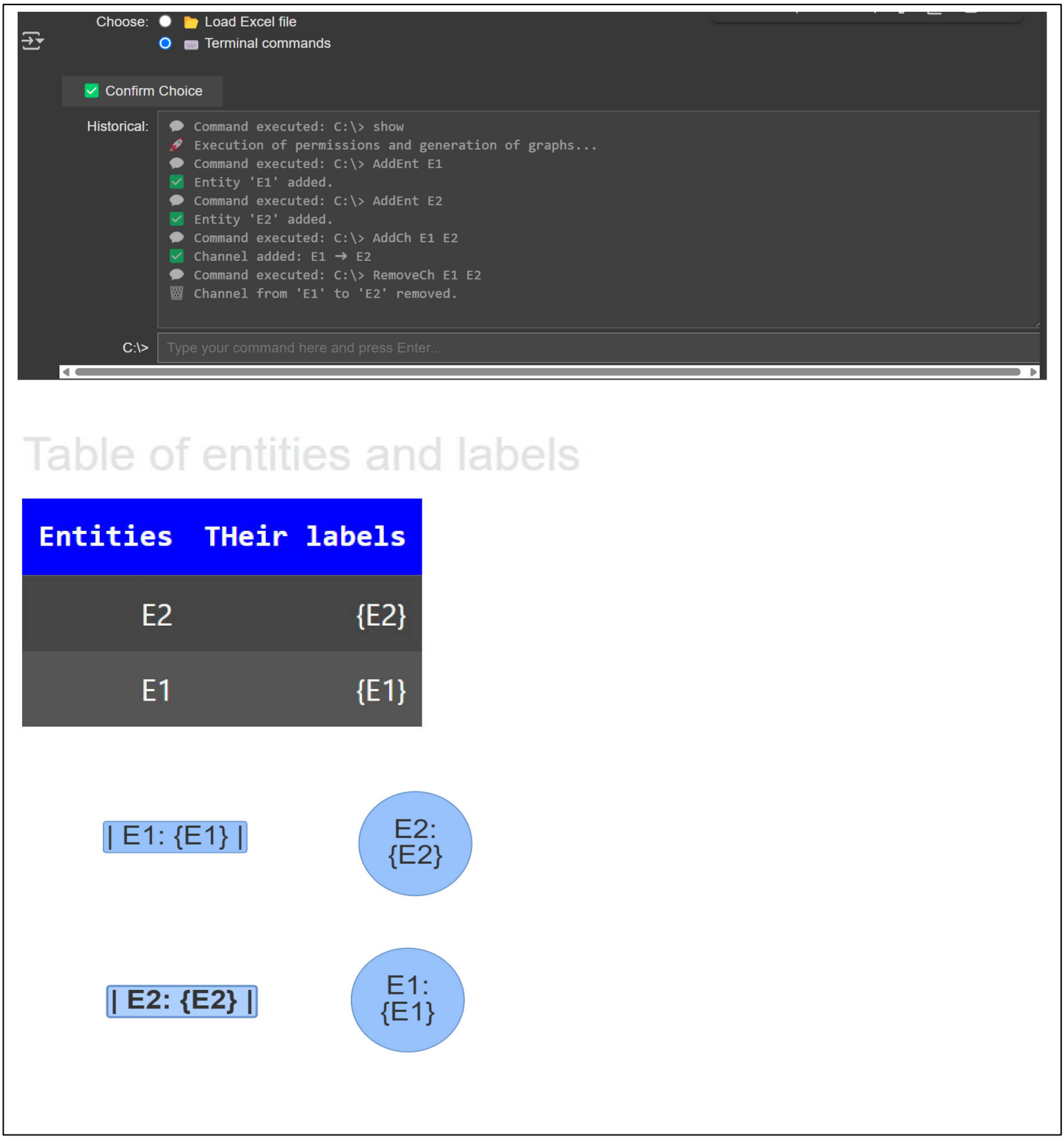


Figure 16: Retrait canal entre *E1* et *E2*

5.1.2. Systèmes de sujets et objets sans rôles (DAC, MAC)

Dans les premières versions de notre système, nous ne différencions pas les entités, toute entité pouvait être à la fois source et cible d'un canal, sans distinction explicite entre sujet et objet. Ce modèle simplifié permettait de représenter les relations de manière uniforme.

Afin d'enrichir le modèle et de mieux représenter les responsabilités respectives, nous avons introduit une différenciation claire :

- Les sujets sont les entités actives, c'est-à-dire celles qui initient les actions et requièrent des permissions.

- Les objets sont les entités passives, qui font l'objet des accès ou des manipulations. Les commandes disponibles sont les suivantes :

Commande	**Exemple**	**Description**
AddSub	*AddSub S1*	Crée le sujet *S1*
AddObj	*AddObj O1*	Crée l'objet *O1*
AddCh	*AddCh S1 R O1*	*S1* a la permission de lecture sur *O1*.
RemoveSub	*RemoveSub S1*	Supprime le sujet *S1*
RemoveObj	*RemoveObj O1*	Supprime l'objet *O1*
RemoveCh	*RemoveCh S1 R O1*	Retrait de canal de lecture *S1* vers *O1*
modifyCh	*modifyCh S1 R O1 S1 W O1*	Modifie la permission de *R* en *W* pour *S1 -> O1*

Tableau 11: Commandes d'exécution (Sujet et Objet)

5.1.2.1. Création et retrait de sujet et objets

Considérons des entités *S1, S2, O1*. Nous allons exécuter la création de ces entités pour montrer la structure d'exécution des commandes et leurs présentations graphiques ainsi que leur tableau d'ordre partiel résultant.

Au début, nous avons notre interface d'accueil dans laquelle il n'y a aucune donnée.

Figure 17: Interface d'accueil

Nous allons maintenant créer un sujet nommé *S1* dont la représentation est ci-dessous dans la Figure 18 en donnant les classes d'équivalences et l'ordre partiel des étiquettes au fur à mesure de chaque création d'entité.

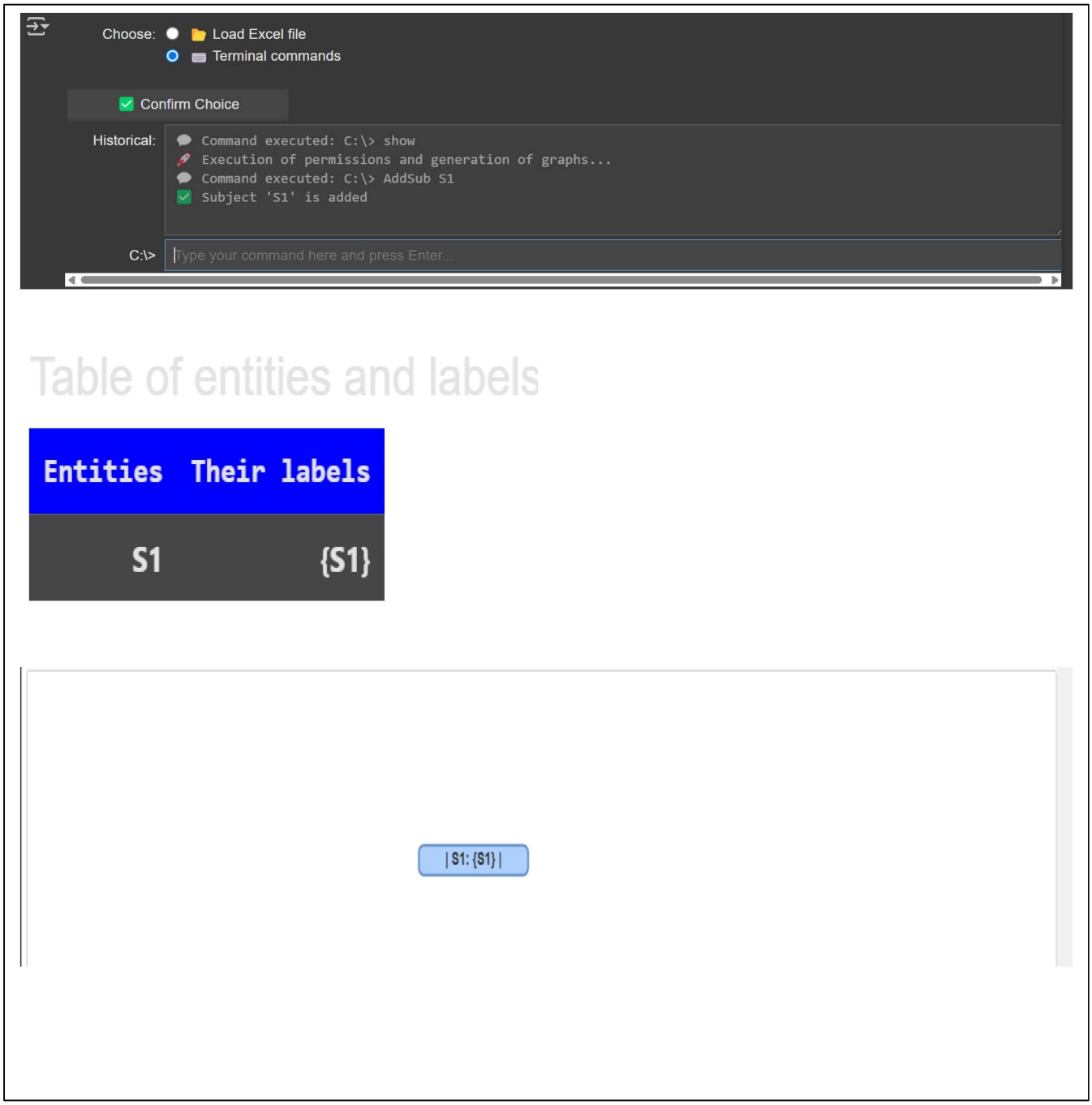


Figure 18: Création d'entité (Sujet *S1*)

Nous allons ajouter aussi un objet nommé *O1* qui sera ajouté dans la Figure 19. L'algorithme calcule les classes d'équivalences et l'ordre partiels des étiquettes au fur à mesure de chaque création d'entité.

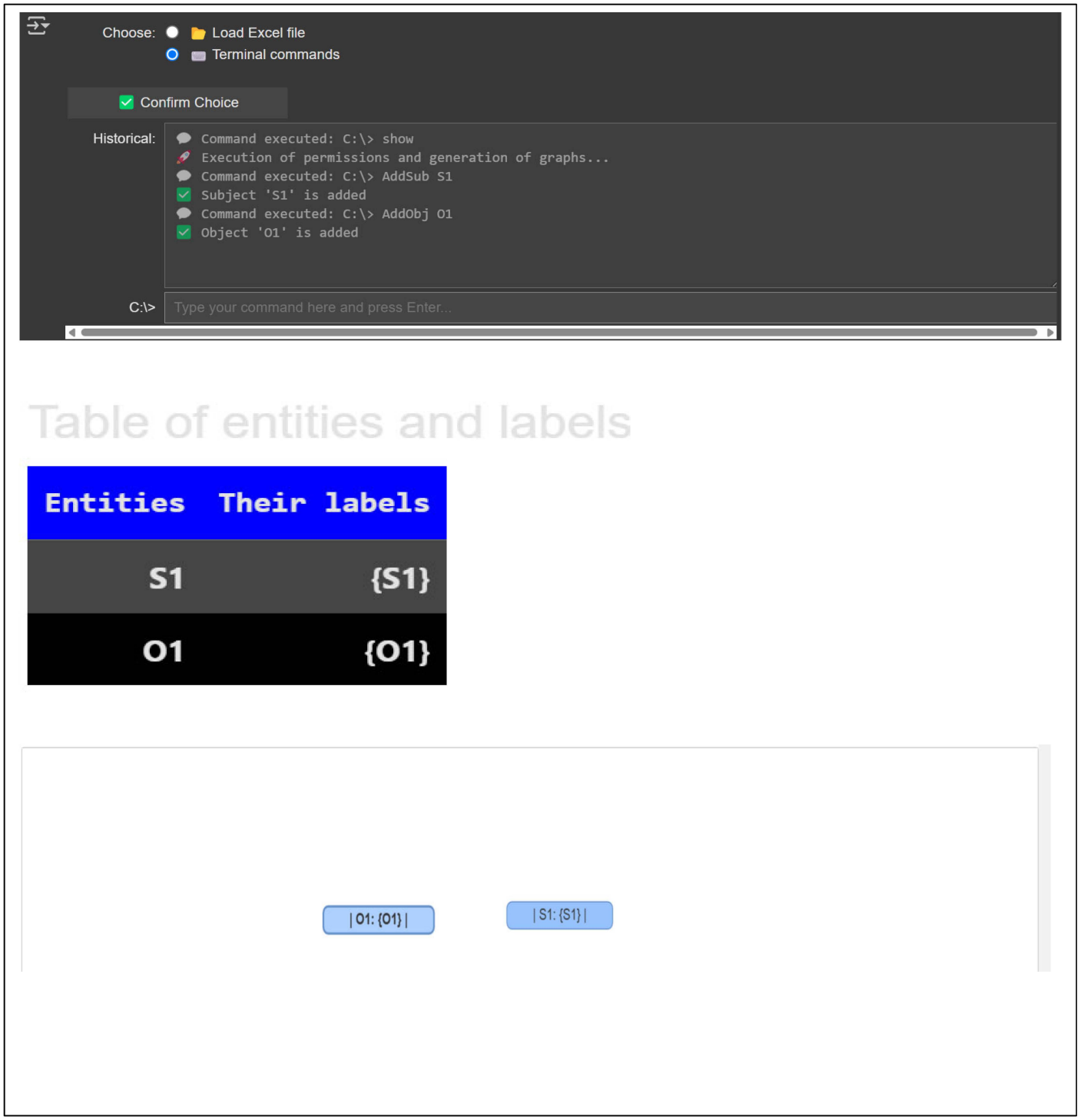


Figure 19: Ajout d'entité (objet *O1*)

La suppression permet de retirer un sujet ou un objet ainsi que l'ensemble des relations qui y sont rattachées. Dans la Figure 20, nous allons retirer le sujet *S1* et l'objet *O1*

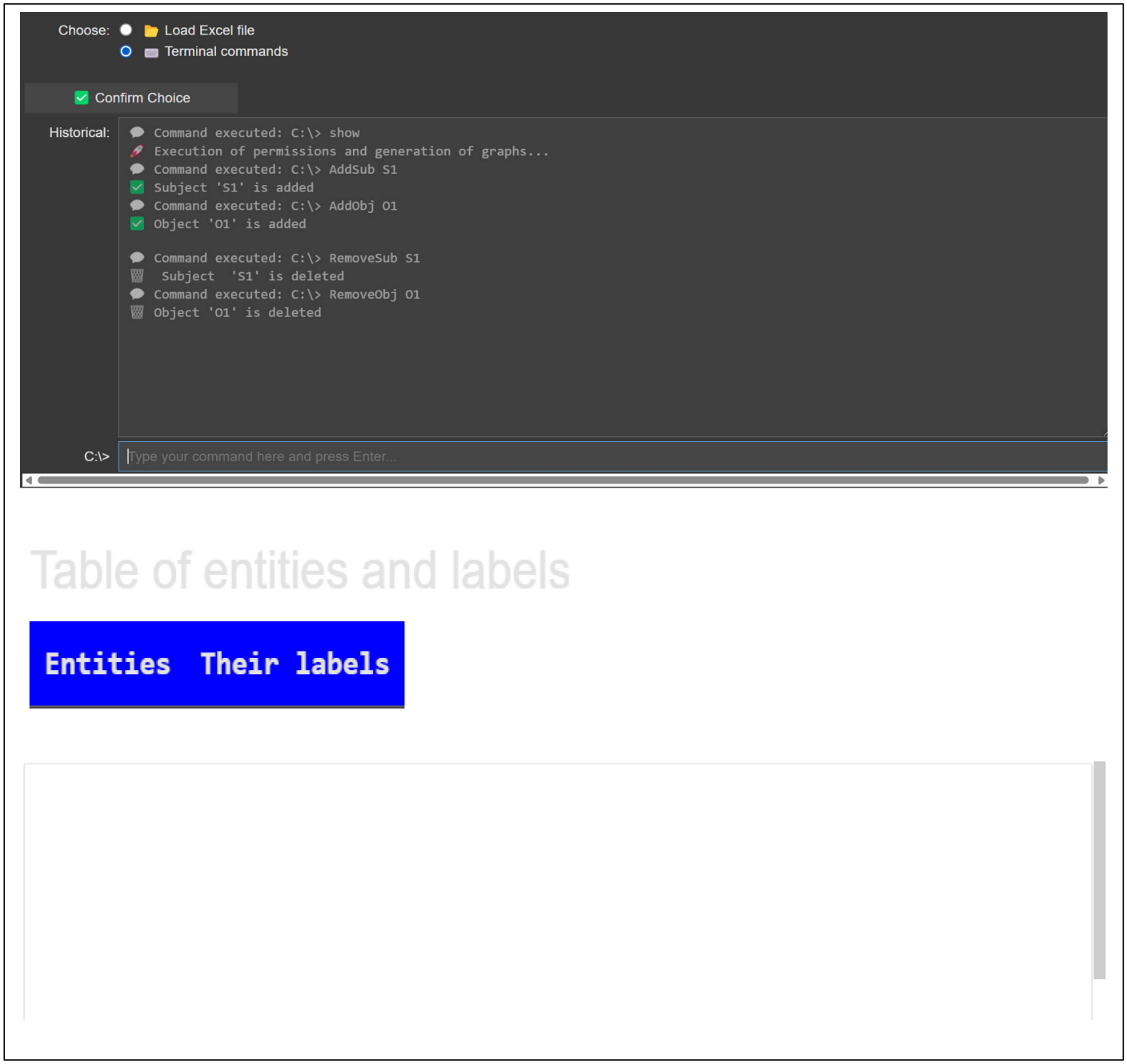


Figure 20: Retrait de l'entité (*S1* et *O1*)

5.1.2.2. Création et retrait de canaux de lecture et écriture

Les canaux représentent les relations d'accès entre les entités : un sujet peut disposer de droits de lecture, d'écriture ou d'autres permissions sur un objet. La création d'un canal consiste à établir explicitement une permission entre une source (le sujet) et une cible (l'objet), avec un type d'accès défini. Nous allons maintenant ajouter des canaux. Une relation indique que *S1* a la permission de lecture *(R)* sur *O1*. Une autre relation indique que *S2* a la permission d'écriture (*W*) sur *O1*.

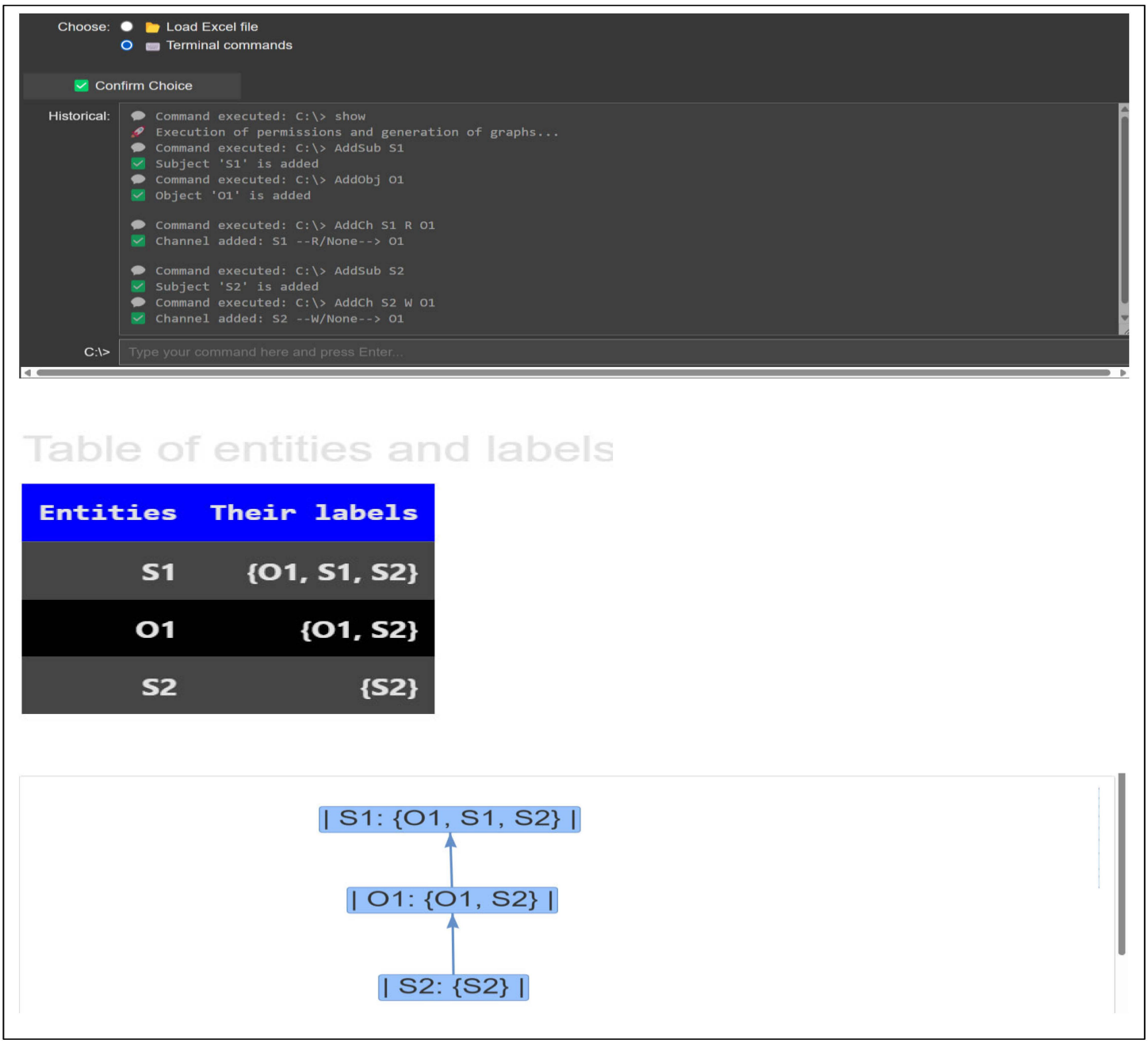


Figure 21: Ajout des canaux des permissions

La suppression d'un canal retire uniquement la relation entre le sujet et l'objet, sans supprimer les entités concernées. Dans la Figure 22, nous retirons la permission de lecture du canal *S1* a *O1*.

Figure 22: Retrait du canal (*S1 R O1*)

5.1.2.3. Discrétionnaire d'Accès (DAC)

Le système traite le modèle DAC (Discretionary Access Control), qui repose sur le principe que les sujets qui créent des objets en deviennent les propriétaires, et ont le droit de contrôler qui peut y accéder. La gestion des permissions est donc laissée à la discrétion du propriétaire de l'objet. Les nouvelles commandes disponibles sont :

Commande	Exemple	Description
AddObj	*S2 AddObj O2*	Création de l'objet *O2* par *S2*
Grant	*S2 Grant S3 O2 R*	Permission de Lecture accordé par *S3* par le propriétaire *S2*

Tableau 12: Commande d'exécution (DAC)

Dans la Figure 23, le sujet S2 crée l'objet O2 et en devient automatiquement le propriétaire.

S3 tente d'accorder la permission de lecture de l'objet O2 au sujet S4 alors qu'il n'est pas le propriétaire de l'objet O2 d'où la raison du message d'erreur.

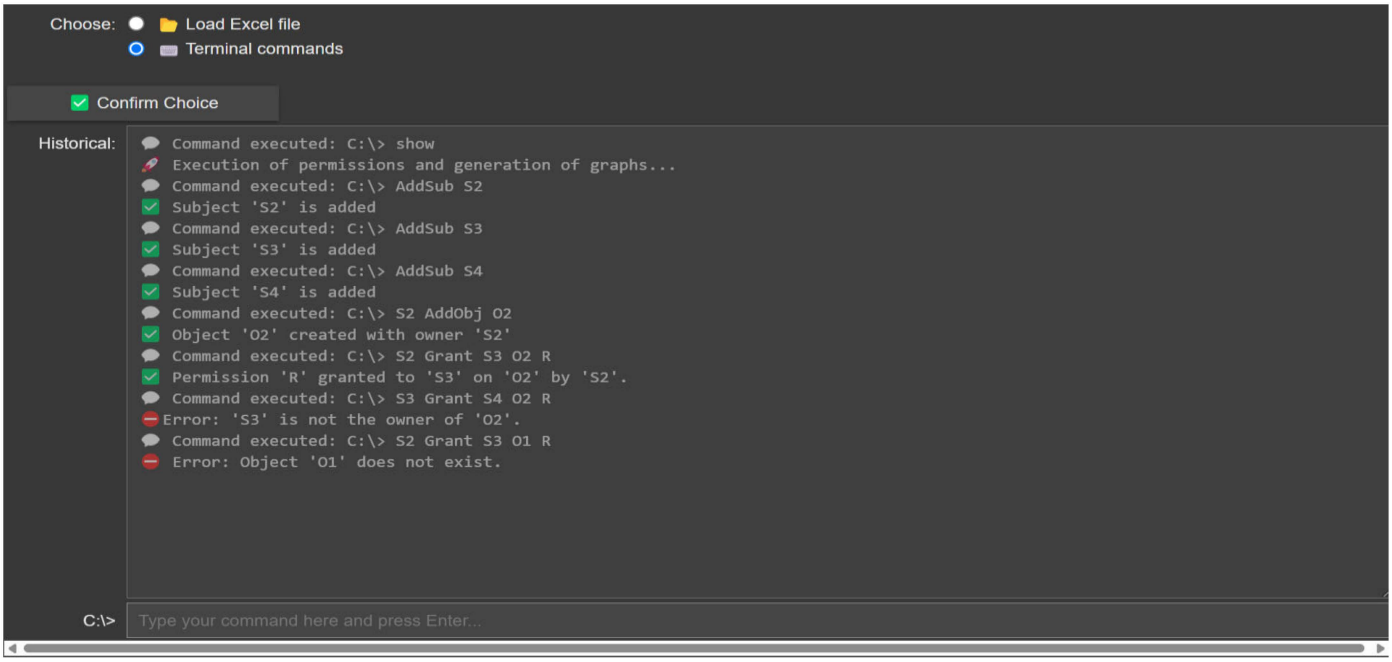


Table of entities and labels

Entities	Their labels
S3	{O2, S3}
O2	{O2}

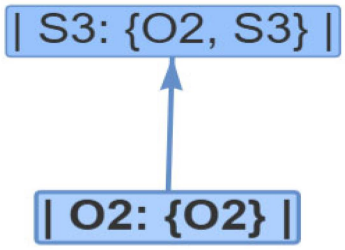


Figure 23: Gestion de propriétaire

5.1.2.4. Modèle d'accès multi-niveaux (MAC)

Comme défini dans la section 2.2.2, dans le modèle MAC, les entités (sujets et objets) sont classées selon différents niveaux de sécurité ou d'intégrité.

Dans l'exemple suivant, nous allons utiliser les options (Fichier Excel et Commande) pour faire les illustrations. Les tableaux que nous utilisons sont des bases de données Excel qui définissent les sujets, les objets et les permissions. L'algorithme exécute cette base de données en choisissant l'option fichier, comme défini dans la Figure 24. Nous importons le fichier contenant les matrices de contrôle d'accès, et l'algorithme extrait les composants fortement connexes (classes d'équivalence) ainsi que l'ordre partiel des étiquettes.

Considérons un réseau avec cinq sujets *S1* à *S5* et quatre objets *O1* à *O4*. Les listes de capacités, *R* (peut lire), ou *W* (peut écrire) sont présentées dans le tableau 11, où la capacité de *S1* est qu'il ne peut écrire que sur *O3*, et ainsi de suite[25].

Subject	Permission	Object
S1	W	O3
S2	R	O1
S2	R,W	O2
S2	R	O3
S3	R	O1
S3	R,W	O3
S3	W	O2
S4	R,W	O2
S4	R,W	O4
S5	R,W	O4

Tableau 13: Matrice de contrôle d'accès de la Figure 24 et 25

Table of entities and labels

Entities	Their labels
O2, O4, S2, S4, S5	{O1, O2, O3, O4, S1, S2, S3, S4, S5}
O3, S3	{O1, O3, S1, S3}
S1	{S1}
O1	{O1}

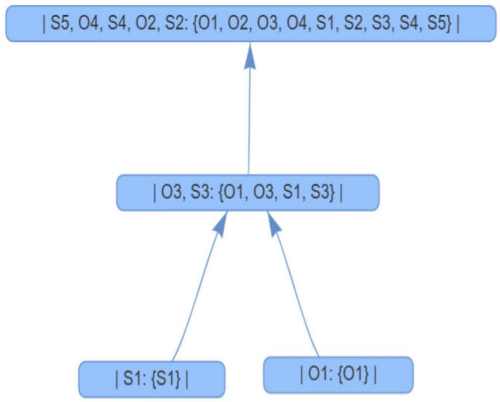


Figure 24: Ordre partiel de matrice de contrôle d'accès du Tableau 12(Option d'exécution Excel)

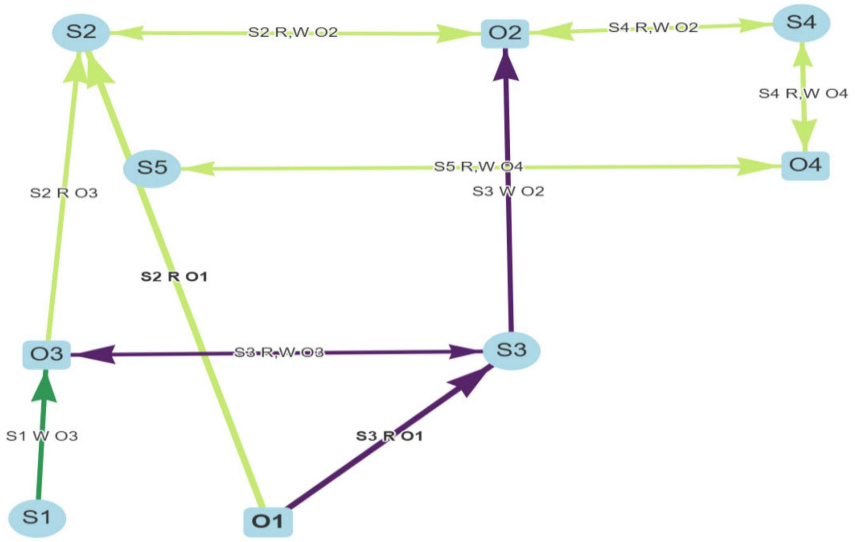
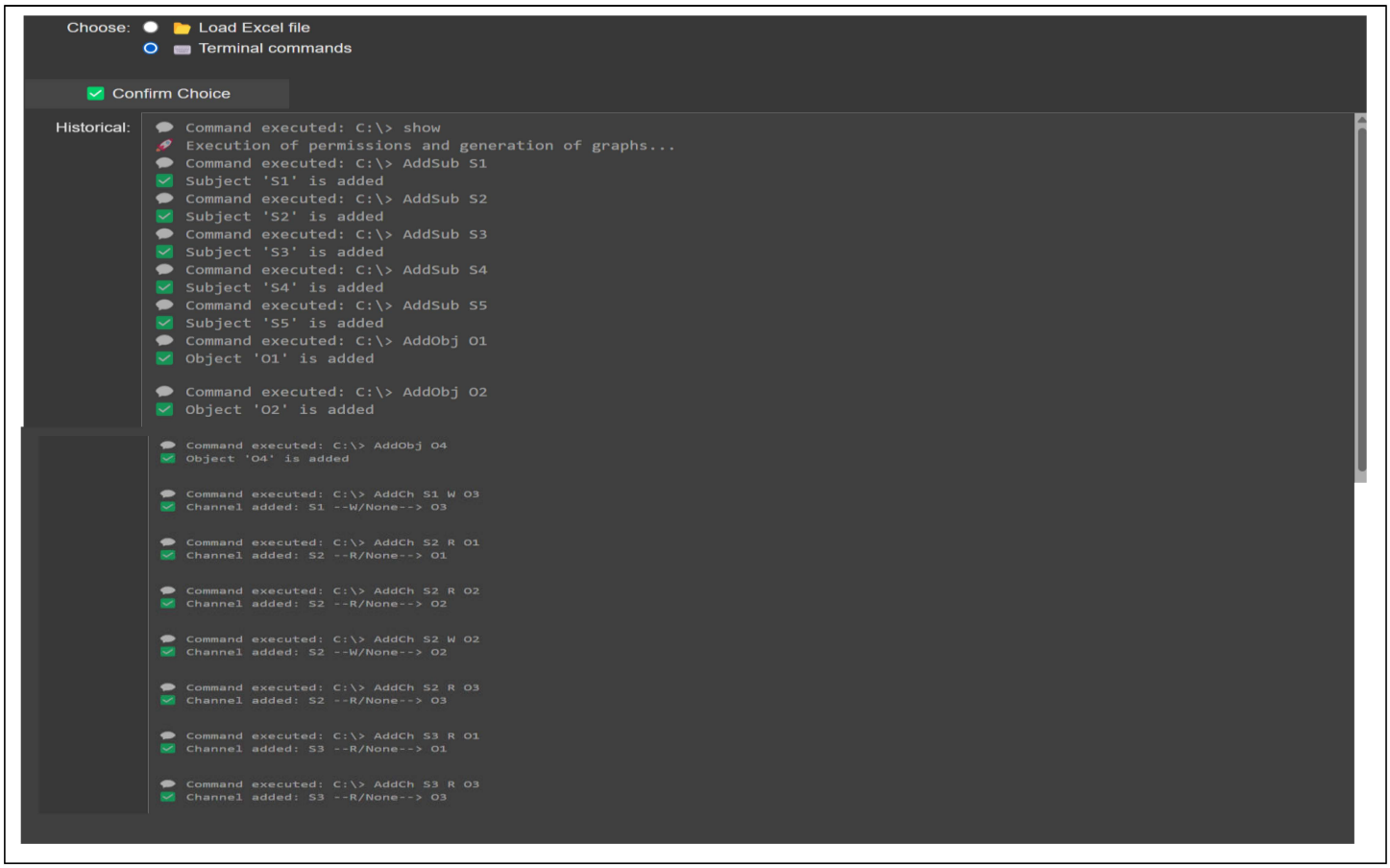


Figure 25: Flux de données des entités et leurs canaux du Tableau 12 et Figure 24

Nous allons illustrer cette approche via commande manuelle dans la Figure 26.



Suite des commandes :

Figure 26: Exécution de commande de la Figure 24 du Tableau 12

La Figure 26 illustre les entités (sujets et objets) avec les étiquettes qui leur sont associées. Ces étiquettes représentent l'ensemble des autorisations et des relations possibles. La colonne de gauche présente les classes d'équivalence, tandis que celle de droite indique l'étiquette attribuée à chaque élément de chaque classe d'équivalence.

Un deuxième exemple plus grand est le suivant. La matrice d'accès est donnée dans le Tableau 13.

Subject	Permission	Object
S1	R	O1
S1	R	O8
S1	R,W	O2
S1	W	O4
S1	W	O6
S2	R	O5
S2	R	O10
S2	W	O7
S3	R	O5
S3	R	O6
S3	R,W	O8
S3	W	O7
S4	W	O3
S5	R	O4
S5	W	O9
S6	R	O1
S6	R	O3
S6	W	O5
S7	R,W	O9
S7	W	O4
S8	R	O5
S8	W	O3

Tableau 14: Listes des matrices contrôle d'accès de la Figure 27

Entities	Their labels
O4, O9, S5, S7	{O1, O2, O3, O4, O5, O6, O8, O9, S1, S3, S4, S5, S6, S7, S8}
O7	{O1, O10, O2, O3, O5, O6, O7, O8, S1, S2, S3, S4, S6, S8}
O2, O6, O8, S1, S3	{O1, O2, O3, O5, O6, O8, S1, S3, S4, S6, S8}
S2	{O1, O10, O3, O5, S2, S4, S6, S8}
O3, O5, S6, S8	{O1, O3, O5, S4, S6, S8}
S4	{S4}
O10	{O10}
O1	{O1}

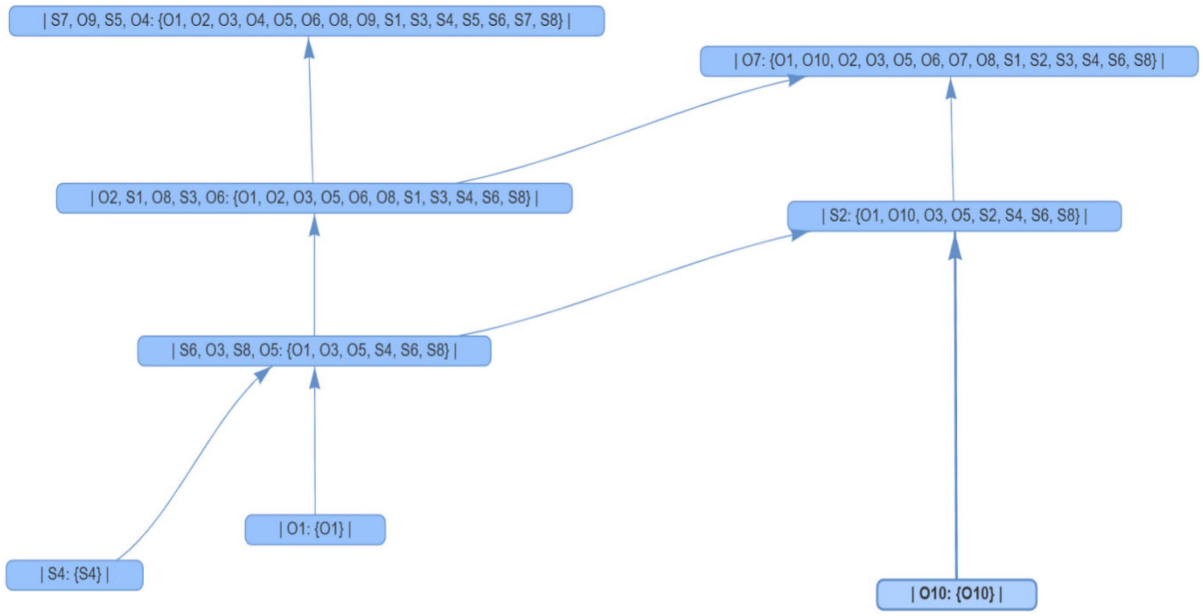


Figure 27: Ordre partiel de matrice de contrôle d'accès du Tableau 13(Option d'exécution Excel)

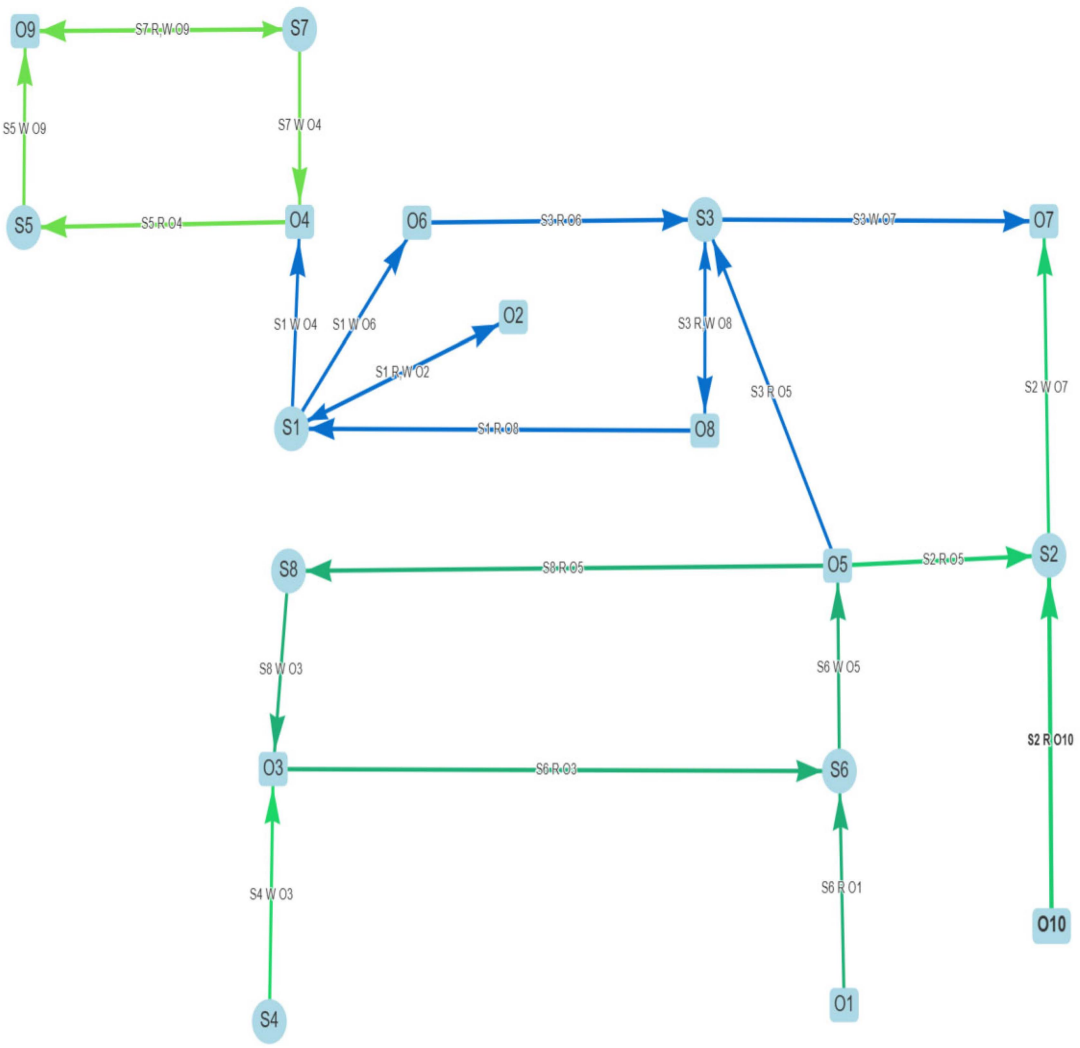


Figure 28: Flux de données des entités et leurs canaux du Tableau 13

Nous pouvons aussi créer une correspondance entre les étiquettes ci-dessus et des étiquettes de sécurité plus conventionnelles. Par exemple {*O1, O2, O3, O4, O5, O6, O8, O9*}, qui est l'étiquette de tous les éléments dans la classe d'équivalence en haut à gauche de la Figure 27, pourrait être appelé « TopSecret 1 », alors que {*O1, O2, O3, O5, O6, O7, O8, O10*}, qui est l'étiquette pour *O7*, pourrait être appelé « TopSecret 2 » et ainsi de suite comme souhaité[25].

Le modèle de la Muraille de Chine a été présenté dans la Section 2.2.2.3. Un exemple typique d'exigence exprimable dans ce système est qu'aucun employé d'une entreprise de consultation

ne peut combiner les données de clients concurrents (par exemple, Banque 1 et Banque 2). Dans notre système, ce type de contrainte est implémenté par l'interdiction de certaines combinaisons d'étiquettes pour certaines entités. Nous avons alors :

- **Restrictions des étiquettes pour les entités** : un employé pourrait se voir restreindre l'accès à certaines étiquettes spécifiques, en fonction de ses responsabilités ou de sa position dans l'organisation. Par exemple, si *BkA* et *BkB* identifient dans les étiquettes les données des Banques *A* et *B*, on pourrait, selon les besoins, défendre les étiquettes contenant le sous-ensemble {*BkA*, *BkB*} pour des entités particulières ou pour tout un réseau. Cela garantirait qu'un employé ne pourrait pas attaquer ou malicieusement accéder aux données des Banques *A* et *B* simultanément.

- Le même mécanisme peut être utilisé pour spécifier des contraintes plus spécifiques que celles traitées par les mécanismes de Muraille de Chine traditionnels. Un exemple serait une contrainte disant que la possibilité de connaître les données d'une compagnie *A* implique la possibilité de connaître les données d'une compagnie *B*, mais pas vice-versa (cette contrainte pourrait être applicable dans le cas d'une situation de dépendance de la compagnie *B* par rapport à la compagnie *A*). Nous avons implémenté des commandes spécifiques pour exprimer ce type de contrainte :

On pourra écrire *Never {A, B, C}* pour exprimer la contrainte que la combinaison *A, B, C* ne peut paraître dans aucune étiquette. On pourra aussi écrire *Never {A, B, C} for {X, Y, Z}*, pour exprimer la contrainte selon laquelle aucune des entités *X, Y* ou *Z* ne pourra jamais avoir des étiquettes contenant les combinaisons *A, B, C*. À noter que la définition usuelle de la Muraille de Chine ne permet pas d'exprimer la contrainte que certaines combinaisons d'étiquettes ne sont permises pour certaines entités spécifiques, autrement dit, la deuxième commande que nous proposons est une nouveauté par rapport à la définition usuelle. Si une commande qui mène à une étiquette exclue est exécutée, un message d'erreur s'affichera. Les nouvelles commandes disponibles sont les suivantes :

Commande	Exemple	Description
AddSub	*AddSub S1*	Crée le sujet *S1*
AddObj	*AddObj O1*	Crée l'objet *O1*
Never	*Never {S1, O1}*	Interdit globalement que *S1* et *O1* apparaissent ensemble dans la même étiquette.
Never	*Never {O1, O2} for {S3}*	Interdiction ciblée, le sujet *S3* ne doit jamais être dans une composante contenante à la fois *O1* et *O2*.
AddCh	*AddCh S1 R O1*	*S1* a la permission de lecture sur *O1*.

Tableau 15: Commandes d'exécution (Interdiction sur les étiquettes)

Nous allons illustrer l'exemple d'une Interdiction globale, *Never {S1, O1}*.

- On crée un sujet *S1* et un objet *O1*.
- La commande *Never {S1, O1}* interdit globalement que *S1* et *O1* apparaissent ensemble dans la même étiquette.
- En exécutant *AddCh S1 R O1*, un lien est créé entre *S1* et *O1*.
- Cela forme une composante qui contient à la fois *S1* et *O1* → **violation de la règle**.
- Le système bloque la commande comme montré dans la Figure 29.

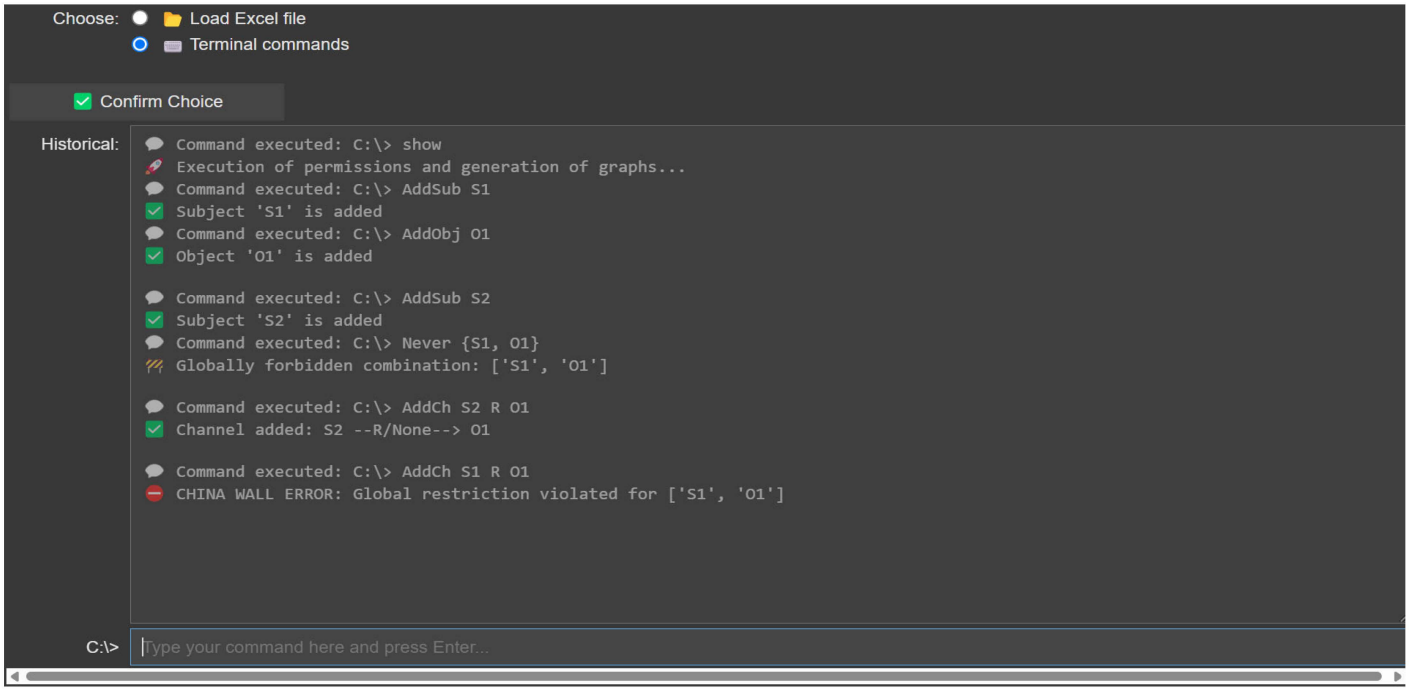


Table of entities and inheritances

Entities	Their labels
S2	{O1, S2}
O1	{O1}

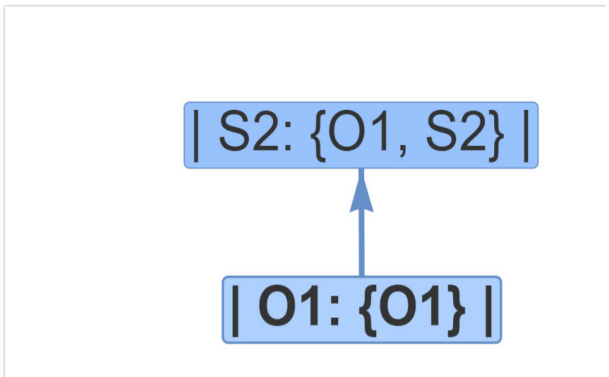


Figure 29: Interdiction globale

Nous allons ensuite présenter un autre exemple avec une interdiction ciblée, Never *{O1, O2} for {S3}*.

- On crée deux objets : *O1* et *O2*.
- Le sujet *S3* **ne doit jamais être dans** une composante contenante à la fois *O1* et *O2*.
- Si l'étiquette *{S3, O1, O2}* est formée.
- Elle viole la règle *Never {O1, O2} for {S3}* → la commande est bloquée comme montré dans la Figure 30.

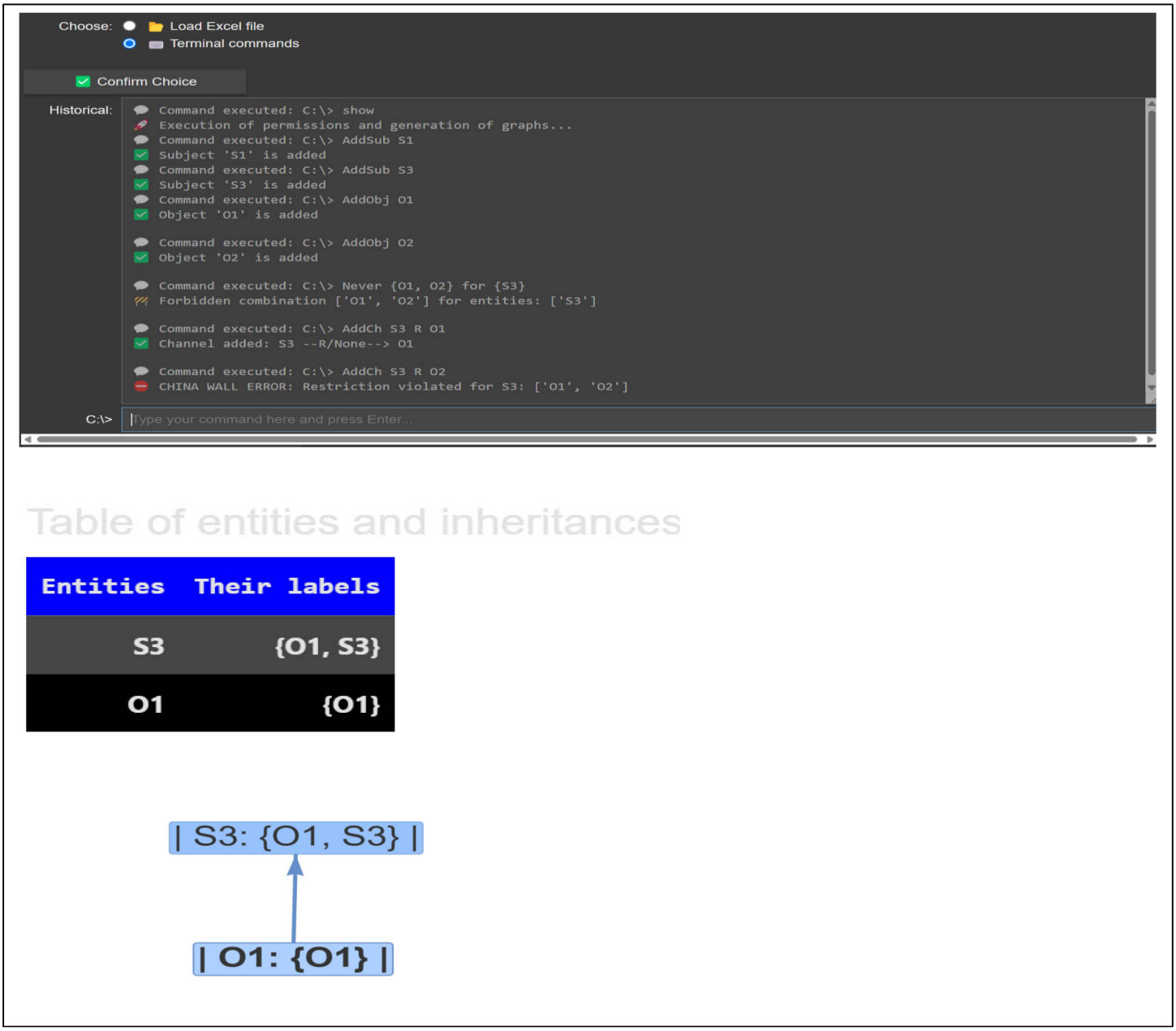


Figure 30: Interdiction ciblée(a)

Nous allons illustrer une deuxième interdiction ciblée : *Never {S1, O1} for {S3, S4, O3}*, il est interdit que *S3*, *S4* ou *O3* contiennent une étiquette incluant à la fois *S1* et *O1*. Si une propagation d'étiquettes entraîne qu'une de ces entités possède *S1* et *O1* dans sa vue d'ensemble, l'ajout de canal est **refusé**. Cette règle protège contre la concentration d'accès simultané à plusieurs ressources conflictuelles dans des entités sensibles.

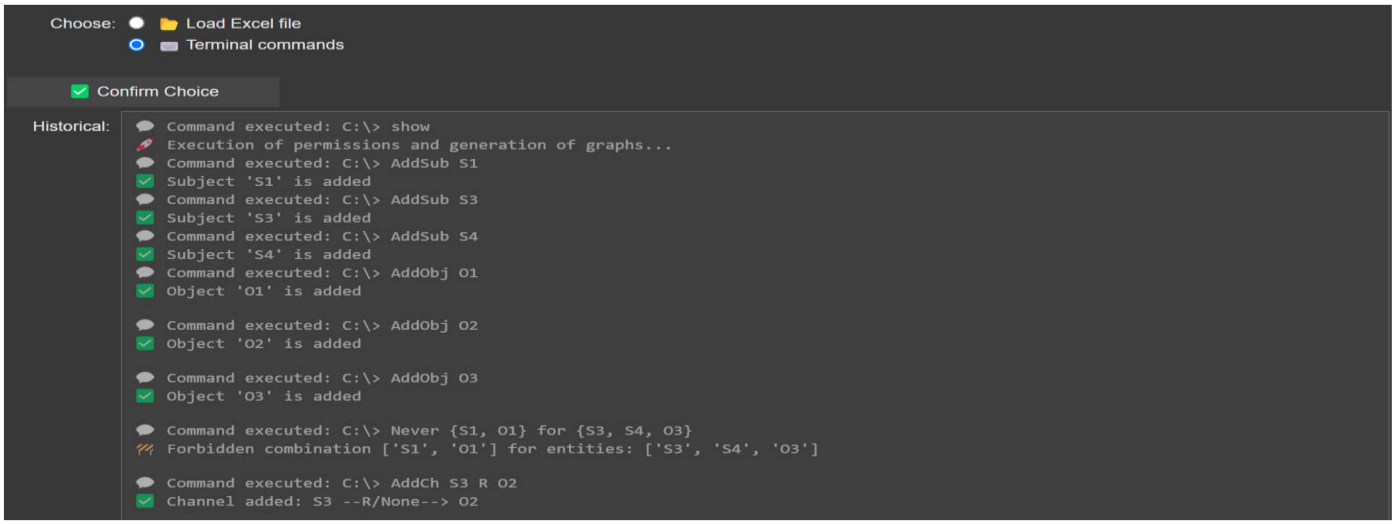


Table of entities and inheritances

Entities	Their labels
S3	{O1, O2, S3}
S4	{O2, S4}
O2	{O2}
O1	{O1}

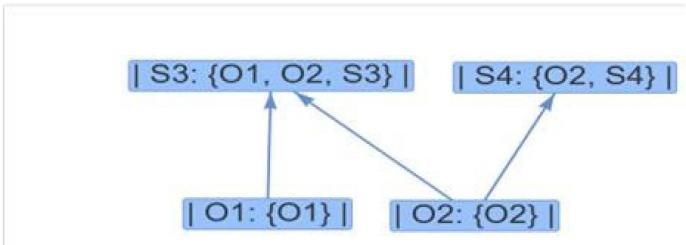


Figure 31: Interdiction ciblée(b)

Comme déjà mentionné, notre implémentation du modèle de la Muraille de Chine va bien au-delà des besoins normalement exprimés dans la littérature, voir Section 2.2.2.3.

5.1.3. Role-Based Access Control (RBAC)

Le modèle RBAC permet de définir les permissions de manière centralisée à travers des rôles. Les entités (ici appelées « sujets ») reçoivent des rôles, et les rôles déterminent les permissions sur les ressources (objets). Les commandes disponibles sont les suivantes :

Commande	Exemple	Description
AddObj	*AddObj O1*	Crée l'objet *O1*
AddRole	*AddRole R1*	Crée rôle *R1*
GrantPermission	*GrantPermission R1 R O1*	Attribue la permission *R* sur *O1* au rôle *R1*
AddSub	*AddSub S1 R1*	Crée un sujet *S1* avec le rôle *R1*
RevokePermission	*RevokePermission R1 W O1*	Retire la permission d'écriture du rôle *R1*
DeassignUser	*DeassignUser S1 R1*	Retire le rôle du sujet *S1* (désassignassions)
ModifyPermission	*ModifyPermission R1 R O1 W*	Modifie la permission *R* sur l'objet *O1* accordée au rôle *R1*, pour qu'elle devienne *W*.
RemoveRole	*RemoveRole R1*	Supprime le rôle *R1*

Tableau 16: Commandes d'exécution (RBAC)

5.1.3.1. Création, retrait de rôles, modification de rôles

Dans la Figure 32, nous créons le rôle *R1* en lui attribuant la permission de lire (*R*) l'objet *O1*. Ensuite, nous créons un sujet *S1* en lui attribuant le rôle *R1*. Étant donné que *S1* possède le rôle *R1*, et que ce rôle lui confère la permission de lire *O1*, *S1* peut donc lire *O1*.

Figure 32:Gestion des rôles (*S1 R O1* avec le rôle *R1*)

Nous allons donner la permission d'écriture au rôle *R1* sur l'objet *O1* comme présenté dans la Figure 33.

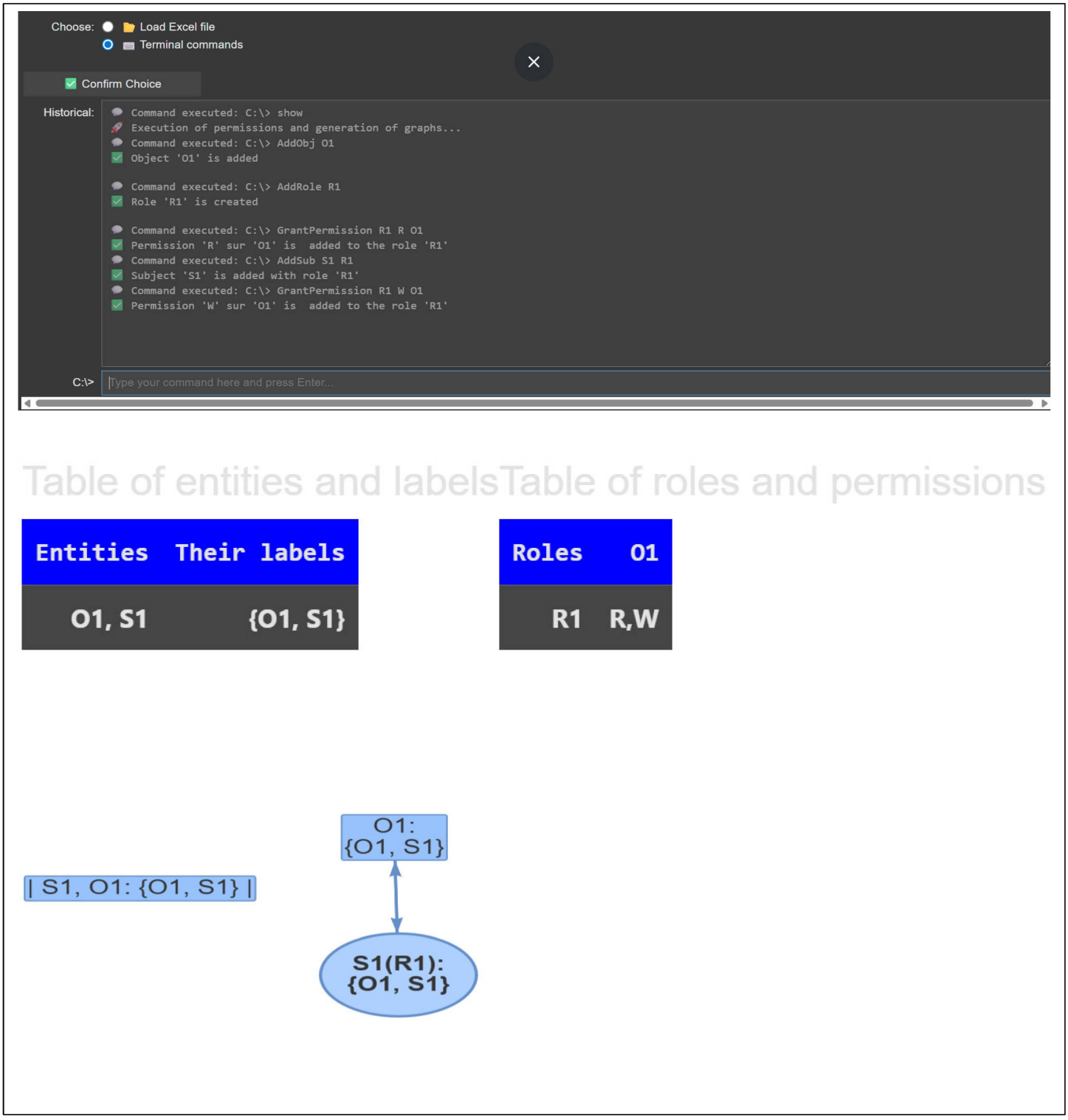


Figure 33:Gestion des rôles (*S1 R*, *W O1* avec le rôle *R1*)

Nous allons maintenant retirer la permission d'écriture (*W*) du rôle *R1* sur l'objet *O1*.

Toutes les instances de cette permission propagées aux sujets ayant le rôle sont retirées. Le rôle *R1* et les autres permissions éventuellement associées restent inchangés.

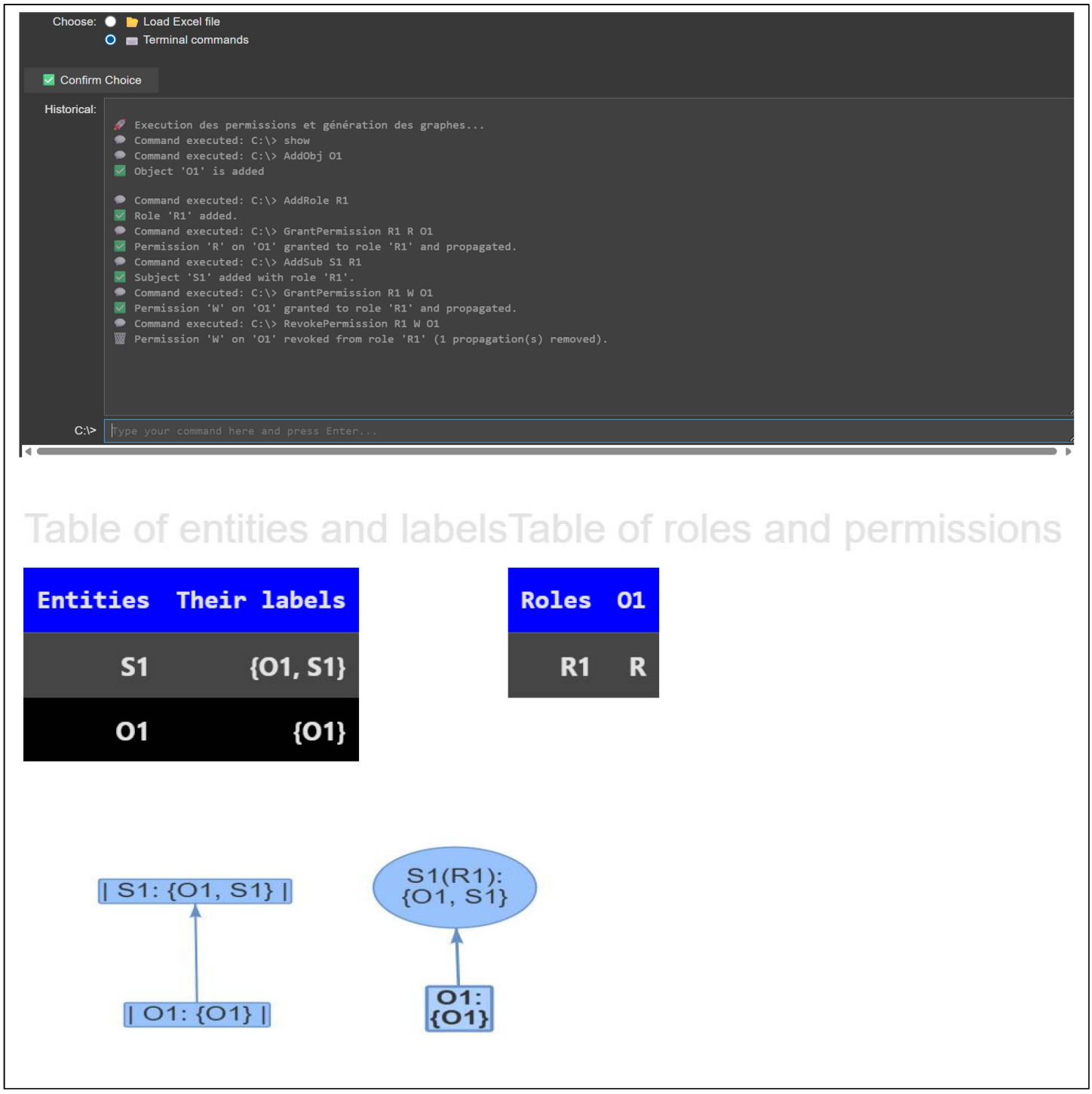


Figure 34:Retrait permission *W* au Rôle *R1*

Nous allons modifier la permission *R* sur l'objet *O1* accordée au rôle *R1*, pour qu'elle devienne *W*, dans ce cas, la permission de lecture est modifiée par la permission d'écriture sur l'objet *O1* accordée au rôle *R1*, cela veut dire que *R1* a seulement la permission d'écrire sur l'objet *O1* comme montré dans la Figure 35.

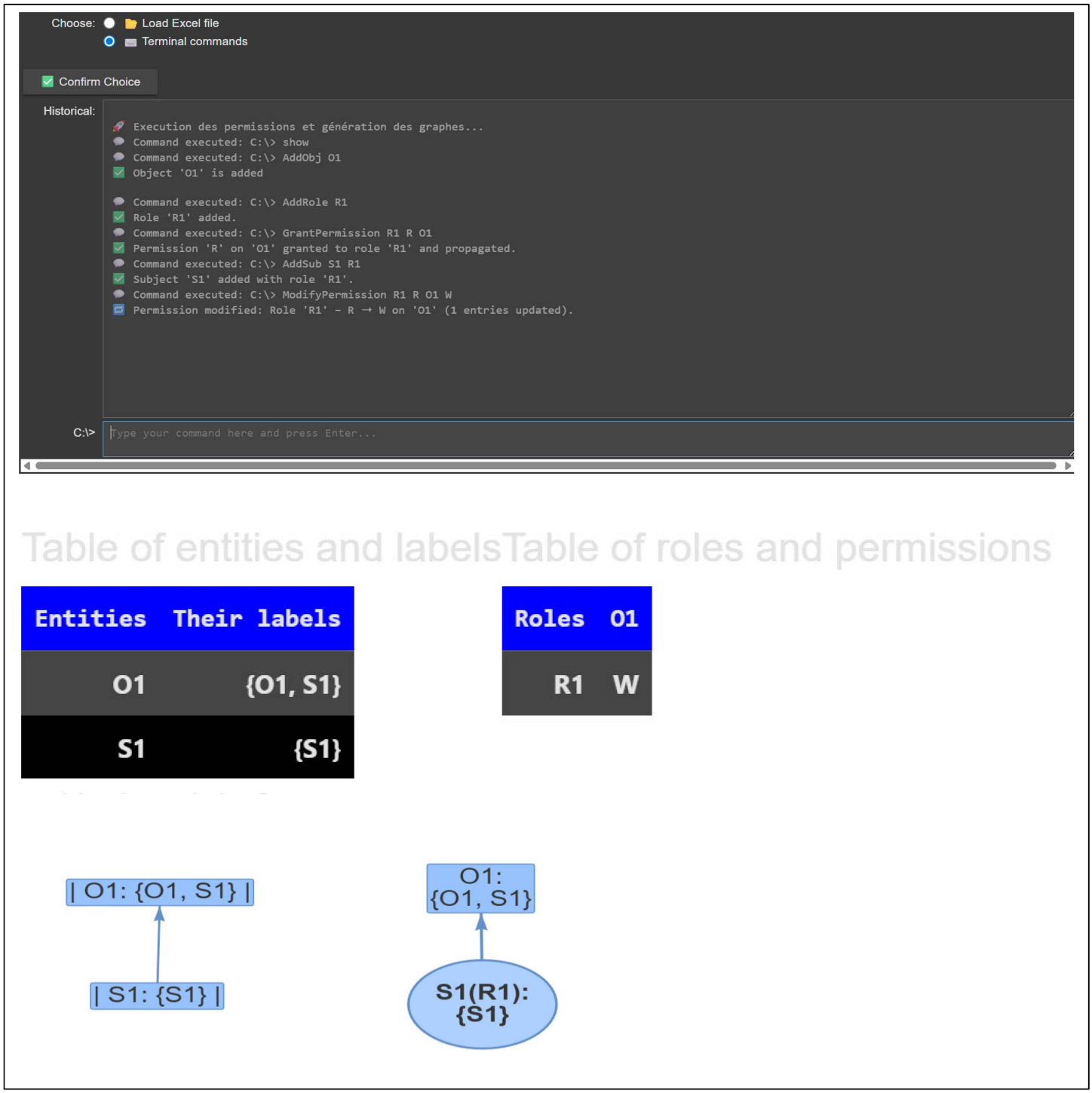


Figure 35: Modification de la permission R sur l'objet O1 accordée au rôle R1

Si un rôle acquiert une permission, toutes les entités ayant ce même rôle doivent bénéficier de cette permission. Comme le montre la Figure 36, nous constatons que *R1* a la permission de lecture (*R*) sur l'objet *O1* et la permission d'écriture (*W*) sur l'objet *O2*. Étant donné que *S1* et *S2* ont tous deux le rôle *R1*, ils disposeront des mêmes permissions sur les objets *O1* et *O2*.

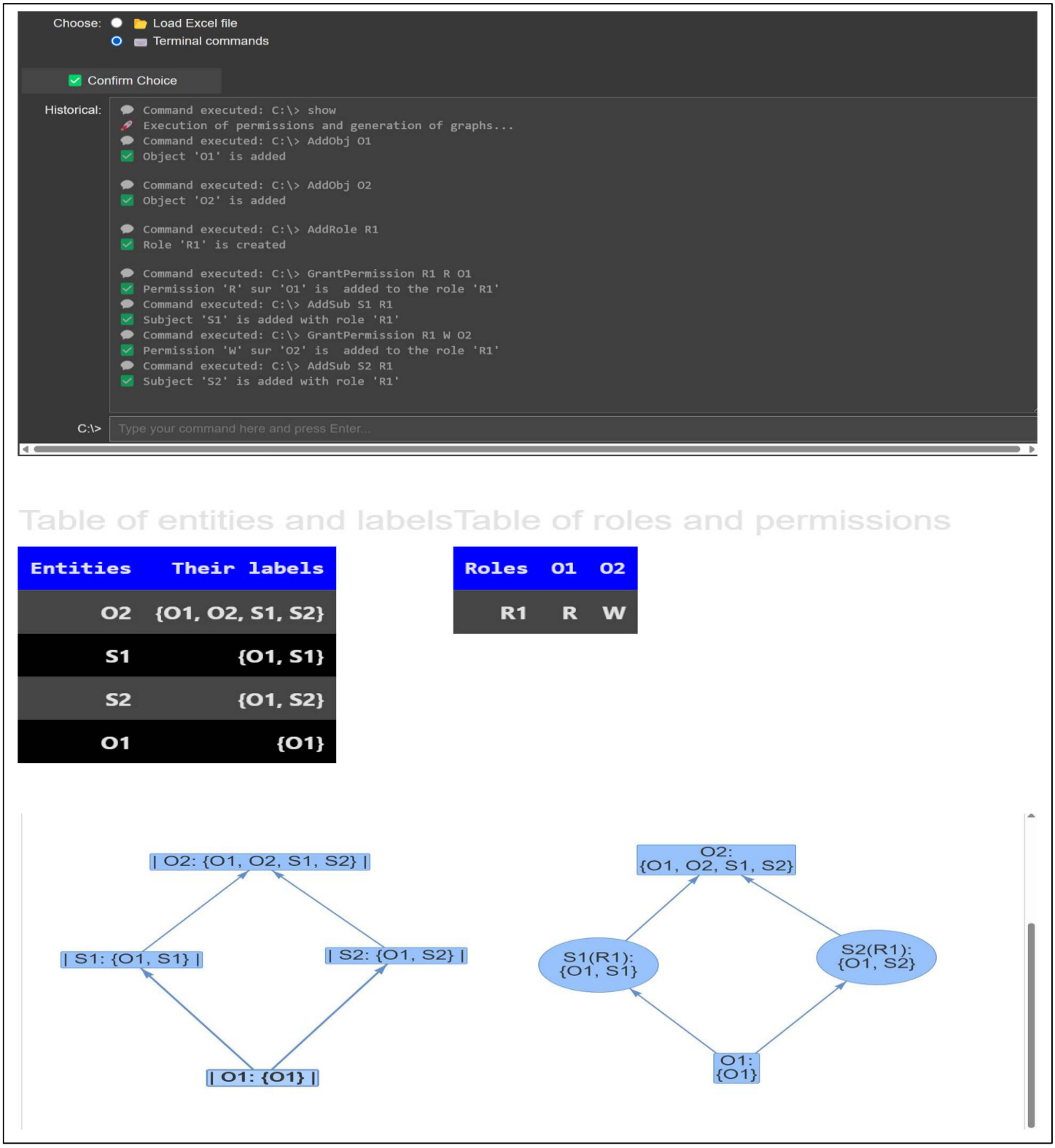


Figure 36:Permission Multiple

5.1.3.2. La gestion d'attribution de rôles

Étant donné des ensembles de sujets et de rôles, nous pouvons analyser les flux de données générés en attribuant différentes combinaisons de rôles aux sujets. Chaque combinaison est une

configuration RBAC différente. Nous suivons dans ce qui suit la présentation de [23].

Les questions auxquelles une telle analyse peut répondre pour une configuration donnée comprennent :

- Comment les données peuvent-elles circuler entre les sujets (et donc leurs utilisateurs) et les objets ?
- Quelles sont les entités les plus secrètes et celles qui ont la plus haute intégrité ?
- Les contraintes de séparation des données spécifiées sont-elles mises en œuvre ?
- Y a-t-il des parties du réseau qui ne semblent pas utiles pour les flux de données de sujet à sujet ou d'utilisateur à utilisateur ?
- Existe-t-il des configurations équivalentes à celle actuelle ?

Bien entendu, afin que notre système puisse effectuer ces analyses, un logiciel approprié doit être développé et certaines difficultés doivent être résolues. Entre autres dans la pratique, les autorisations des systèmes RBAC peuvent être conditionnelles [23].

Pour établir une relation avec des recherches connexes, nous utilisons un exemple tiré d'un article précédent de Radhika et al sur un sujet étroitement lié [29], qui à son tour a été inspiré par des exemples contenus dans un article de Chakraborty et al sur le minage de rôles pour passer de RBAC à ABAC [30] . Trois objets et quatre rôles sont proposés, voir Tableau 16. Lorsque les sujets sont affectés à des rôles, nous obtenons la fonction de RBAC *SP* qui indique quels canaux existent entre les sujets et les objets (Définition 6), qui sera utilisée pour l'analyse.

Subject	Permission	Object	Role
S1	R	O1	R1
S1	W	O3	R1
S2	W	O2	R2
S3	R	O3	R3
S4	R	O1	R4
S4	R	O3	R4

Tableau 17: Matrice de contrôle d'accès Rôle et Permissions de la Figure 37

Le système crée un graphique représentant les entités, les relations entre ces entités et leurs autorisations. Dans ce cas précis, chaque sujet, objet et relation est transformé en un nœud et une arête dans un graphe.

Le système applique ensuite le modèle RBAC pour centraliser et structurer ces autorisations. Chaque entité se voit attribuer une étiquette représentant l'ensemble des permissions dont elle dispose, en tenant compte des rôles attribués et des règles de transitivité. Les résultats sont ensuite visualisés sous forme de graphe et de tableau d'étiquettes.

Nous présenterons seulement quelques possibilités, pour illustrer l'analyse qui peut être menée sur différents sujets et combinaisons de rôles, ainsi que les résultats très différents qui peuvent être obtenus. Mais on voit tout de suite dans la Figure 37 que :

1. L'objet *O1*, pouvant être lu uniquement, sera toujours de la plus haute intégrité dans cette configuration ; il peut être interprété comme un objet contenant des valeurs constantes.
2. L'objet *O2*, ne pouvant être qu'écrit, sera toujours de la plus haute confidentialité.
3. Puisque les rôles *R3* et *R4* n'ont que des autorisations de lecture, les sujets n'ayant que ces rôles auront le plus grand secret ; d'autre part, puisque le rôle *R2* n'a que des autorisations en écriture, les sujets n'ayant que ce rôle aura la plus haute intégrité.

Une question de principe peut s'exprimer à propos de cette table de rôles, puisque toutes les attributions possibles de rôles aux sujets conduiront à des flux commençant par *O1,* qui doivent alors avoir des contenus constants, et se terminant par *O2* (pas nécessairement dans le même flux), qui ne peut pas être modifié, être lu par n'importe quel sujet, donc par n'importe quel utilisateur. On peut répondre à des questions de ce type en disant que le système pourrait permettre d'autres configurations, avec des flux différents. Par exemple, *O1* dans cette configuration peut contenir des données écrites dans une configuration précédente, et *O2* peut être la destination des données à lire dans des configurations futures. De telles considérations intéresseront un concepteur ou un administrateur de systèmes.

Voyons quelques exemples de combinaisons des sujets et rôles. La première combinaison considère une configuration avec quatre sujets, chacun ayant l'un des quatre rôles.

Le flux de données global étiqueté est donné sur la Figure 37 et son graphique d'ordre partiel.

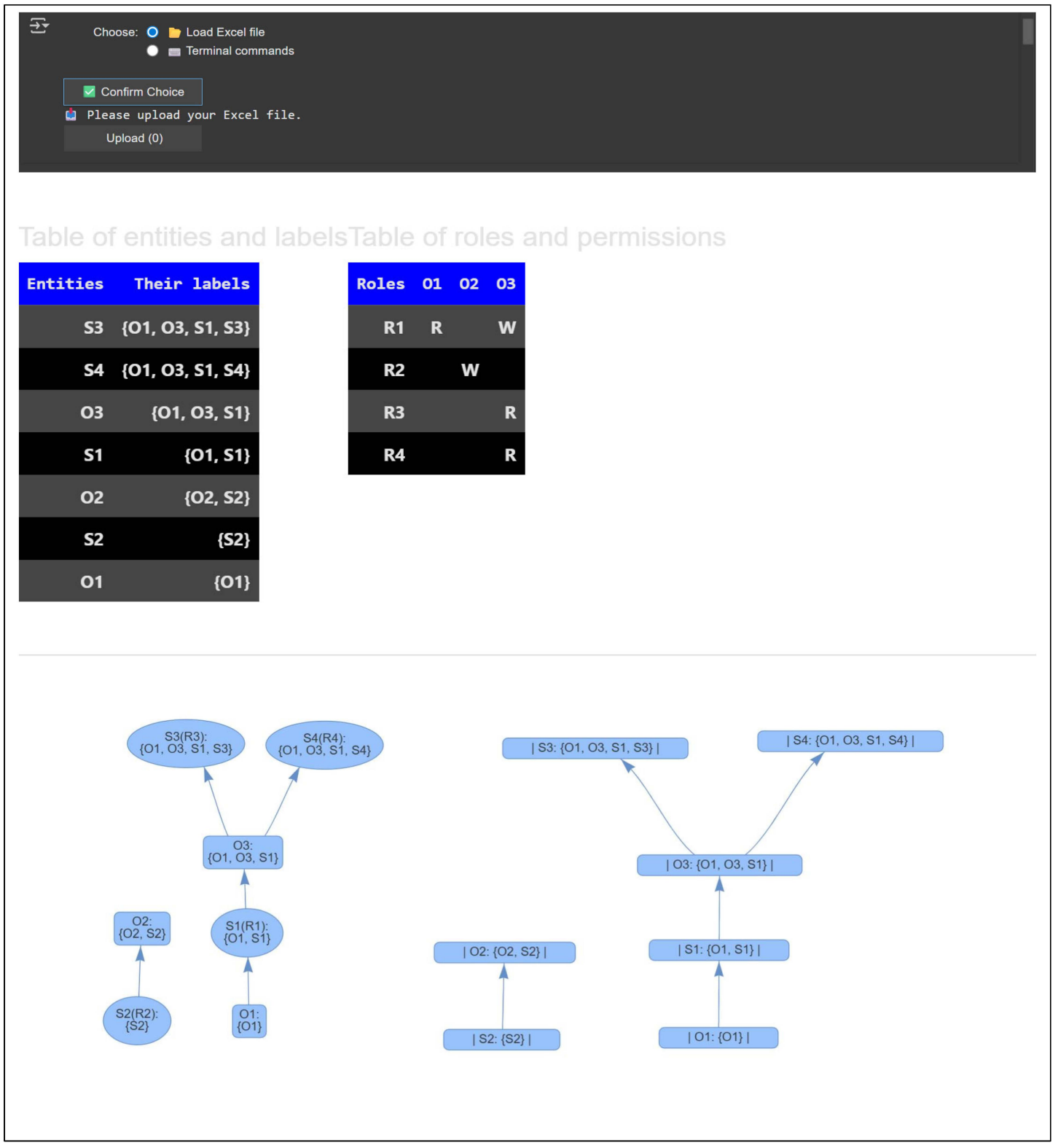


Figure 37 : Un réseau avec un rôle par sujet et Son ordre partiel

Comme mentionné dans la Figure 37, l'autorisation de *S4* de lire à partir de *O1* n'est pas affichée puisque *S4* peut obtenir les données de *O1 indirectement par transitivité.* Il n'existe pas d'entités équivalentes par paire. Cette attribution de rôle met en œuvre une situation de conflit ou de séparation de données (éventuellement une *muraille de Chine* [31],[32]) entre les données de *O2* et celles de *O1,O3* , puisqu'aucune entité ne peut avoir d'étiquettes contenant *{O1,O2}* ou *{O2 ,O3}* . Le flux est alors divisé en deux. Nous voyons que *S2* peut écrire des données sur *O2* (imaginez une base de données de statistiques qui est en cours de compilation pour être utilisée dans des configurations ultérieures). Le flux de droite pourrait être interprété comme *S1* traitant des données provenant d'un utilisateur, utilisant des données constantes dans *O1* et écrivant les résultats dans *O3,* résultats qui peuvent être consommés par les utilisateurs associés à *S3* et *S4.*

Le résultat obtenu après l'exécution du fichier Excel par l'algorithme représente aussi deux tableaux dans la Figure 37, le premier tableau illustre les entités (sujets et objets) avec les étiquettes qui leur sont associées. Ces étiquettes représentent l'ensemble des autorisations et des relations possibles.

L'ordre partiel des classes d'équivalence détermine le niveau de confidentialité et d'intégrité de chaque entité. Les étiquettes associées aux entités indiquent quelles données peuvent être transmises à quelles entités, ainsi que l'origine des données qui peuvent atteindre chaque entité. De plus, ces étiquettes peuvent être exploitées pour définir des tables de routage garantissant que les données soient acheminées exclusivement vers les entités autorisées à les recevoir [25].

Le système affiche également la table des rôles, qui montre les permissions accordées à chaque rôle pour un objet donné. Le rôle *R1* a la permission de lire l'objet *O1* et d'écrire sur l'objet *O3*. *R2* a la permission d'écrire *O2*. *R3* a la permission de lire *O3*. Enfin, R4 a la permission de lire *O1* et de lire aussi *O3*. Ces permissions accordées aux rôles sont affectées aux sujets comme montré dans le Tableau 16.

Nous allons illustrer cette approche via commande manuelle présenter dans la Figure 37.

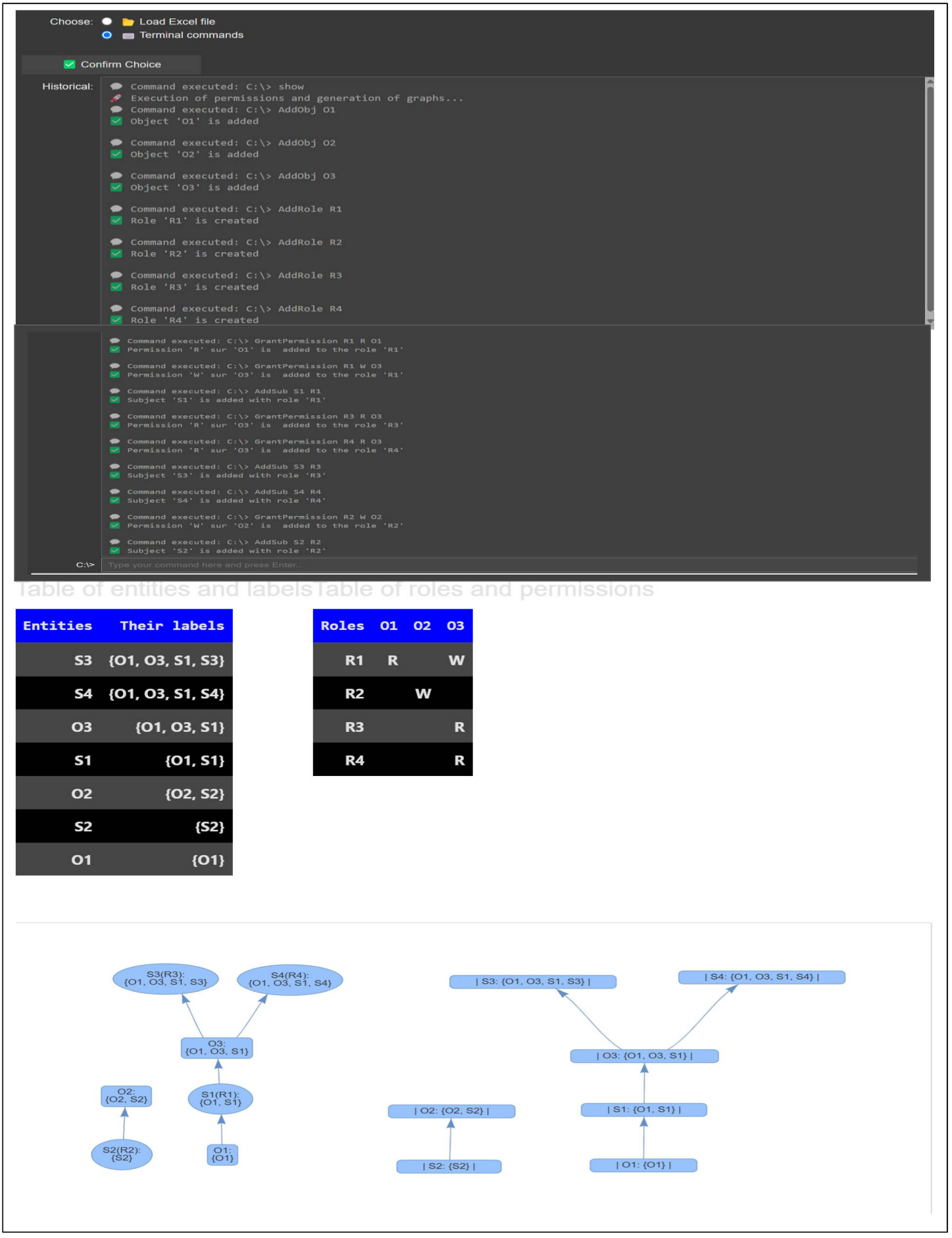


Figure 38: Exécution de commande de la Figure 37 du tableau 16

5.1.4. Attribute-Based Access Control (ABAC)

Le modèle ABAC, quant à lui, apportera une dimension plus fine à la gestion des accès, en permettant de définir des politiques d'accès basées sur des attributs spécifiques des utilisateurs, des ressources, et des environnements d'exécution.

Avec le modèle ABAC, nous pouvons utiliser des **étiquettes** pour définir les règles de permission dans le ***PAP*** (Policy Administration Point). Ces règles seront évaluées par le ***PDP***, (Policy Decision Point). Par exemple, considérons *A* et *B* comme des entités :

- A peut lire les données de B si et seulement si l'étiquette B est un sous-ensemble de l'étiquette de *A*, $Lab\ (B) \subseteq Lab\ (A)$.
- A peut écrire des données sur B si et seulement si l'étiquette de A est incluse dans l'étiquette de *B*, $Lab\ (A) \subseteq Lab\ (B)$.

Ce mécanisme repose sur l'attribution d'attributs spécifiques, dans ce cas des étiquettes, aux sujets et aux objets, ce qui permet d'établir des relations d'accès dynamiques et contextuelles entre les différentes entités du système.

L'intégration de ces deux modèles, en plus de ceux déjà présents dans notre système, permettra de couvrir un plus large éventail de scénarios d'utilisation tout en améliorant la sécurité et la gestion des autorisations dans des contextes de plus en plus hétérogènes et dynamiques.

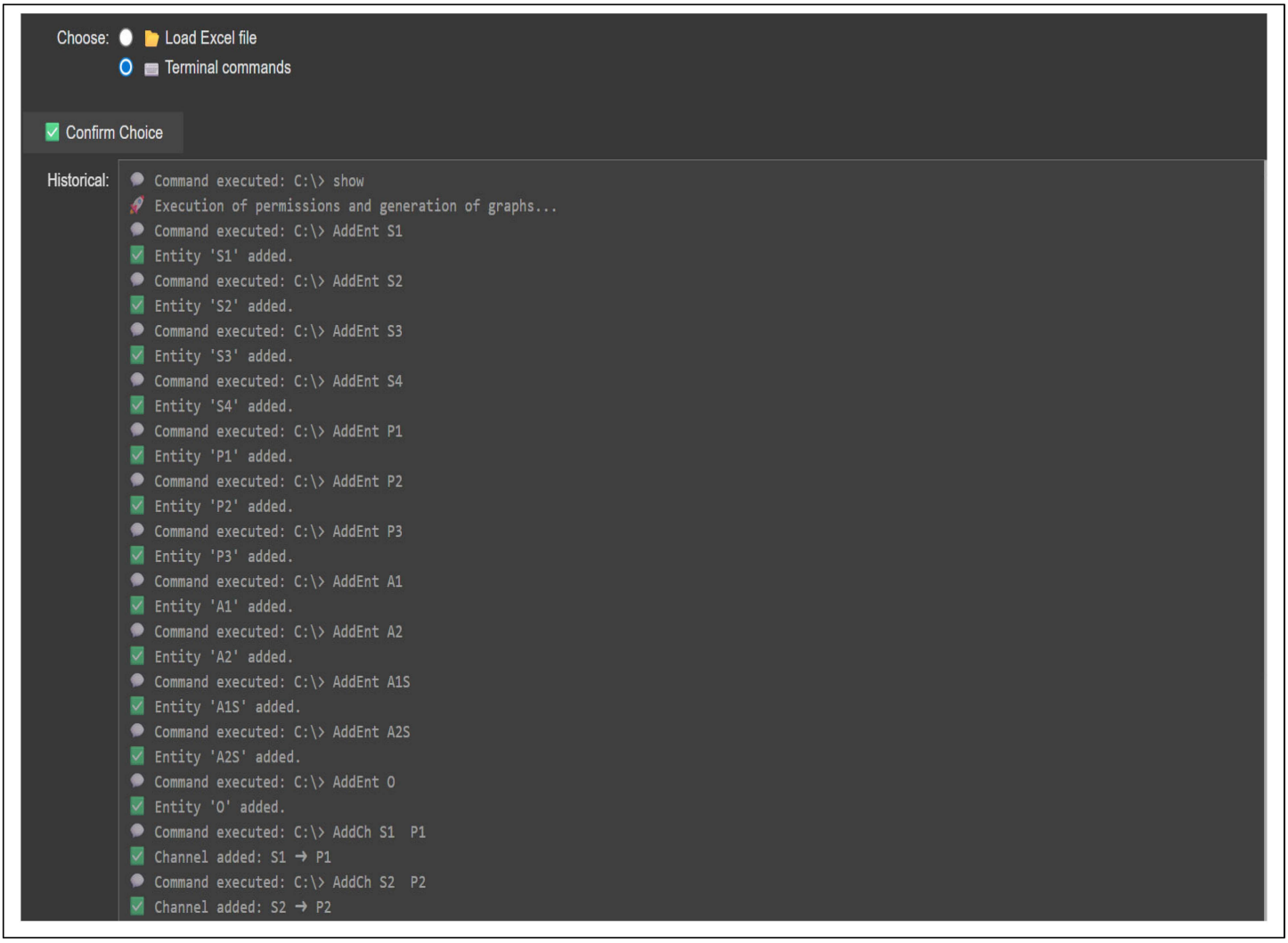
Nous n'avons pas encore implémenté le modèle ABAC dans notre système. Cependant, nous donnerons les résultats attendus utilisant les opérations que nous avons implémentées.

Dans une entreprise, nous avons :

- Quatre points de vente: *S1*, *S2*, *S3*, *S4*.
- Deux centres de traitement de données : *P1* et *P2*.
- Deux centres d'analyse de données, *A1* et *A2*.
- Un centre de traitement des commandes, *O*.

La politique de sécurité requiert l'existence de deux flux de données distincts : l'un pour les données commerciales, l'autre pour les données statistiques. Pour le premier flux, les politiques de flux sont les suivantes :

- Les données de vente provenant de *S1* vont à *P1*
- Les données de vente provenant de *S2* vont à *P2*
- Les données de vente provenant de *S3* et *S4* vont vers *P3*
- *P1* et *P2* peuvent partager des données librement
- *P3* peut envoyer des données à *P1* et *P2*
- *P1* et *P2* envoient des données à *A1* et *A2*
- *A1* et *A2* ne peuvent pas obtenir les résultats du traitement de l'autre.



```
Channel added: S2 → P2
Command executed: C:\> AddCh S3  P3
Channel added: S3 → P3
Command executed: C:\> AddCh S4  P3
Channel added: S4 → P3
Command executed: C:\> AddCh P1  P2
Channel added: P1 → P2
Command executed: C:\> AddCh P2  P1
Channel added: P2 → P1
Command executed: C:\> AddCh P3  P2
Channel added: P3 → P2
Command executed: C:\> AddCh P2  A1
Channel added: P2 → A1
Command executed: C:\> AddCh P2  A2
Channel added: P2 → A2
Command executed: C:\> AddCh A1S  O
Channel added: A1S → O
Command executed: C:\> AddCh A1S  A2S
Channel added: A1S → A2S
```

Table of entities and labels

Entities	Their labels
A2	{A2, P1, P2, P3, S1, S2, S3, S4}
A1	{A1, P1, P2, P3, S1, S2, S3, S4}
P1, P2	{P1, P2, P3, S1, S2, S3, S4}
P3	{P3, S3, S4}
A2S	{A1S, A2S}
O	{A1S, O}
S1	{S1}
S3	{S3}
S4	{S4}
A1S	{A1S}
S2	{S2}

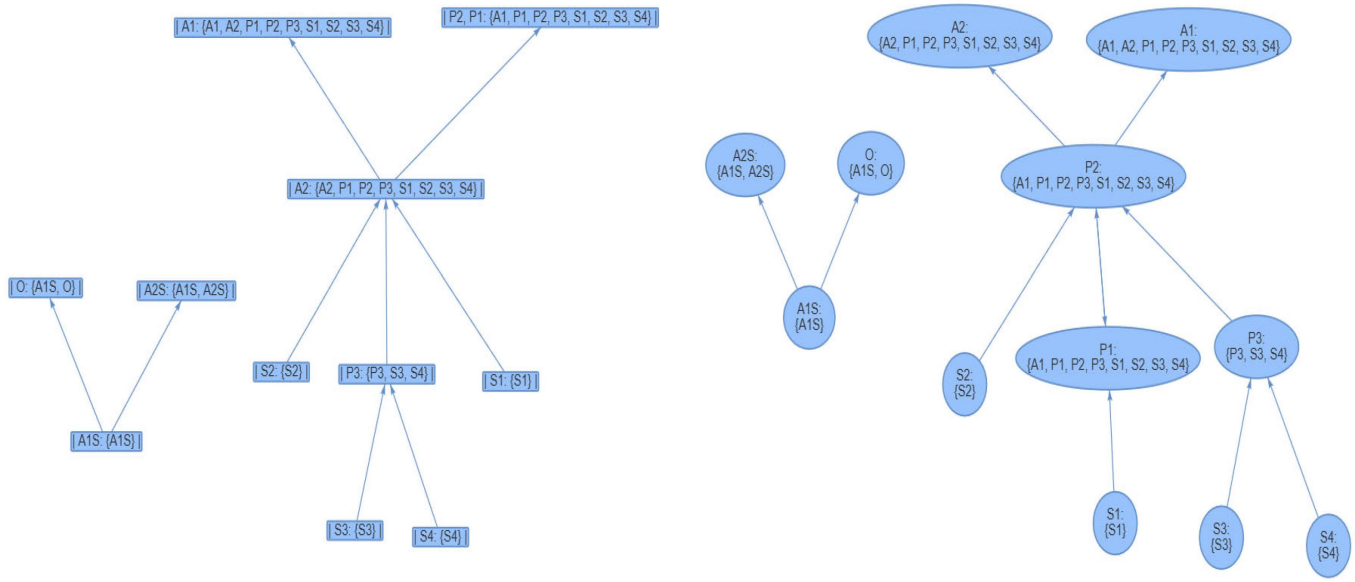


Figure 39:Gestion des attributs (ABAC)

Dans ce flux de données, notez la classe d'équivalence {*P1*, *P2*}, qui constitue un élément unique dans l'ordre partiel, ainsi que la politique de la « muraille de Chine » entre A1 et A2.

Dans le deuxième flux de données, nous supposons l'existence d'un programme fiable qui compile les statistiques à partir des données de *A1*, incluant les niveaux de stock et excluant les données d'identification telles que les noms des points de vente. Les résultats sont transmis à une partie distincte de *A1*, appelée *A1S*, qui les transmet à une partie distincte de *A2*, appelée *A2S*, et à *O*. Il s'agit donc d'un flux de données distinct, qui démarre dans *A1S* et implique des données différentes.

5.1.5. Introduction de Sous-Systèmes Graphiques et de Boîtes Conteneurs

En raison de la croissance très rapide des données et des autorisations, notre système doit être conçu pour gérer des grands réseaux tout en restant compréhensible et facile à manipuler. Pour répondre à cette contrainte, nous avons mis en place des **boîtes conteneurs** et de **hiérarchies de conteneurs** :

- **Boîtes conteneurs** : dans des systèmes de taille réaliste, toutes les entités ne pourront pas être affichées simultanément. Pour pallier ce problème, nous avons développé un mécanisme de boîtes conteneurs. Ces boîtes pourront contenir des ensembles d'entités connectées des différentes manières. Elles fourniront un moyen de **c**acher certains sous-systèmes, et de les afficher à la demande.

- **Hiérarchie de conteneurs** : nous introduirons une structure hiérarchique de ces sous-systèmes, où les boîtes pourront contenir d'autres boîtes ou sous-graphes comme montré dans le graphe du figure 41 et 42 sous forme des boutons. Par exemple, en cliquant sur une boîte principale (comme un Hôpital), le système présente graphiquement tous les sous-systèmes associés (comme les bureaux de l'administration, les urgences, etc.). Sélectionnant des sous-systèmes (par exemple les urgences), on affiche les sous-graphes associées (par exemple la base de données des infirmières ou celle des médecins).

Exemple : illustration du graphe avec plusieurs entités

Dans l'exemple illustré par la Figure 40, nous présentons un graphe complexe de flux de données comprenant plusieurs entités interconnectées. Chaque nœud représente soit un sujet (*S*) soit un objet (*O*), et chaque arc correspond à un canal de communication ou une permission (lecture, écriture, ou combinaison des deux). À mesure que de nouvelles permissions ou relations sont ajoutées, la densité du graphe augmente, ce qui rend la visualisation globale plus difficile. La complexité croissante se manifeste par la présence de multiples chemins reliant différentes entités et formant des composantes fortement connexes.

Afin de faciliter l'analyse et la compréhension de ces interactions, nous avons conçu un mécanisme de navigation par sous-graphes. Chaque bouton affiché sur l'interface correspond à une composante connexe détectée automatiquement par l'algorithme de Tarjan. En sélectionnant un bouton spécifique, l'utilisateur peut explorer la sous-structure du graphe associée à cette composante, visualiser l'ensemble des entités qui la composent et examiner l'ordre partiel des classes d'équivalence identifiées.

Les Figures 41 et 42 montrent des exemples de sous-graphes détaillés générés à partir du graphe principal. Chaque sous-graphe met en évidence les relations internes entre les entités de la composante sélectionnée, ainsi que les permissions associées à chaque canal. Cette fonctionnalité permet une meilleure lisibilité et offre à l'administrateur une vue ciblée sur les flux de données spécifiques à chaque cluster d'entités, tout en conservant la possibilité de revenir à la vue d'ensemble du graphe principal.

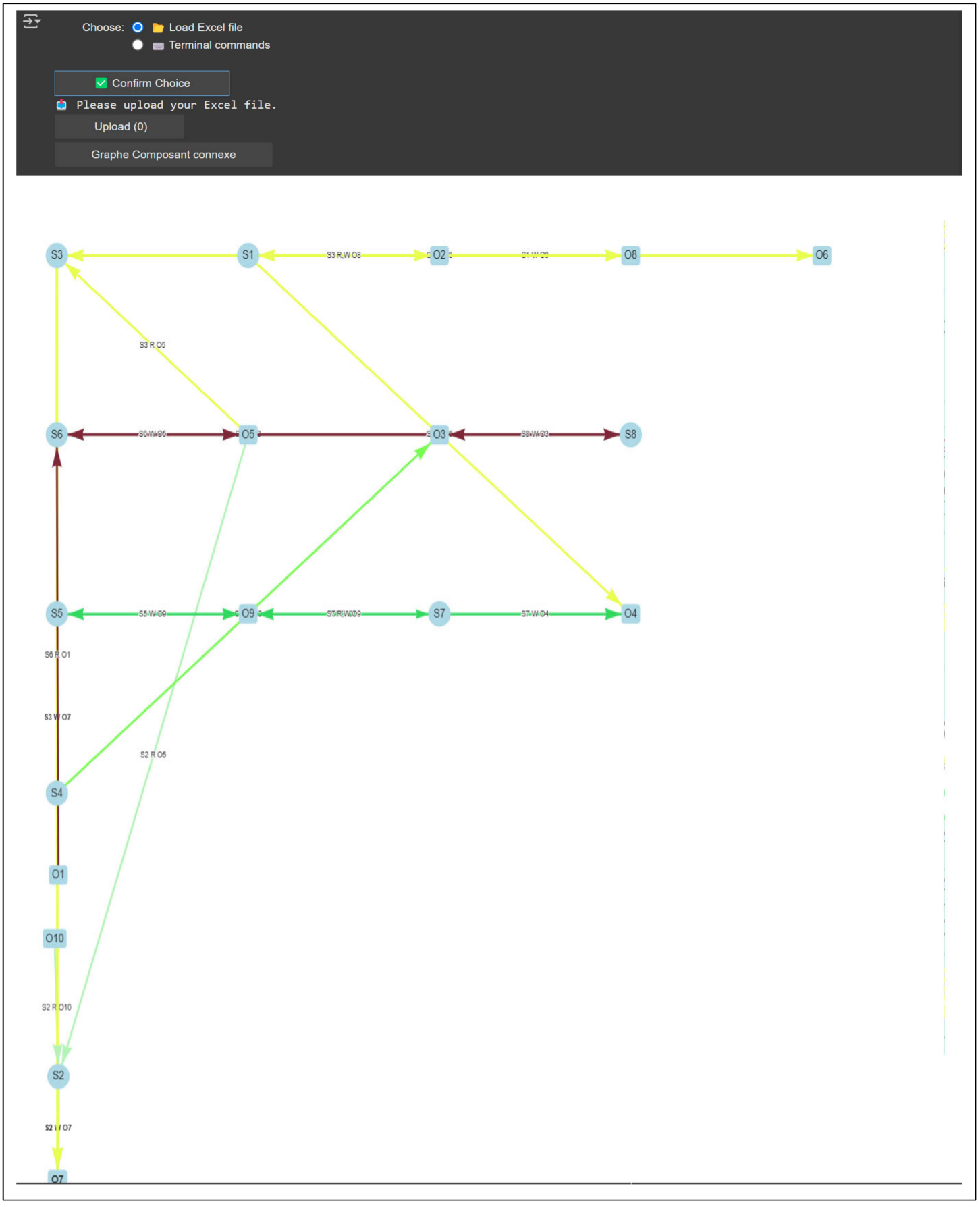


Figure 40: Graphe principale (a)

Suite du graphe

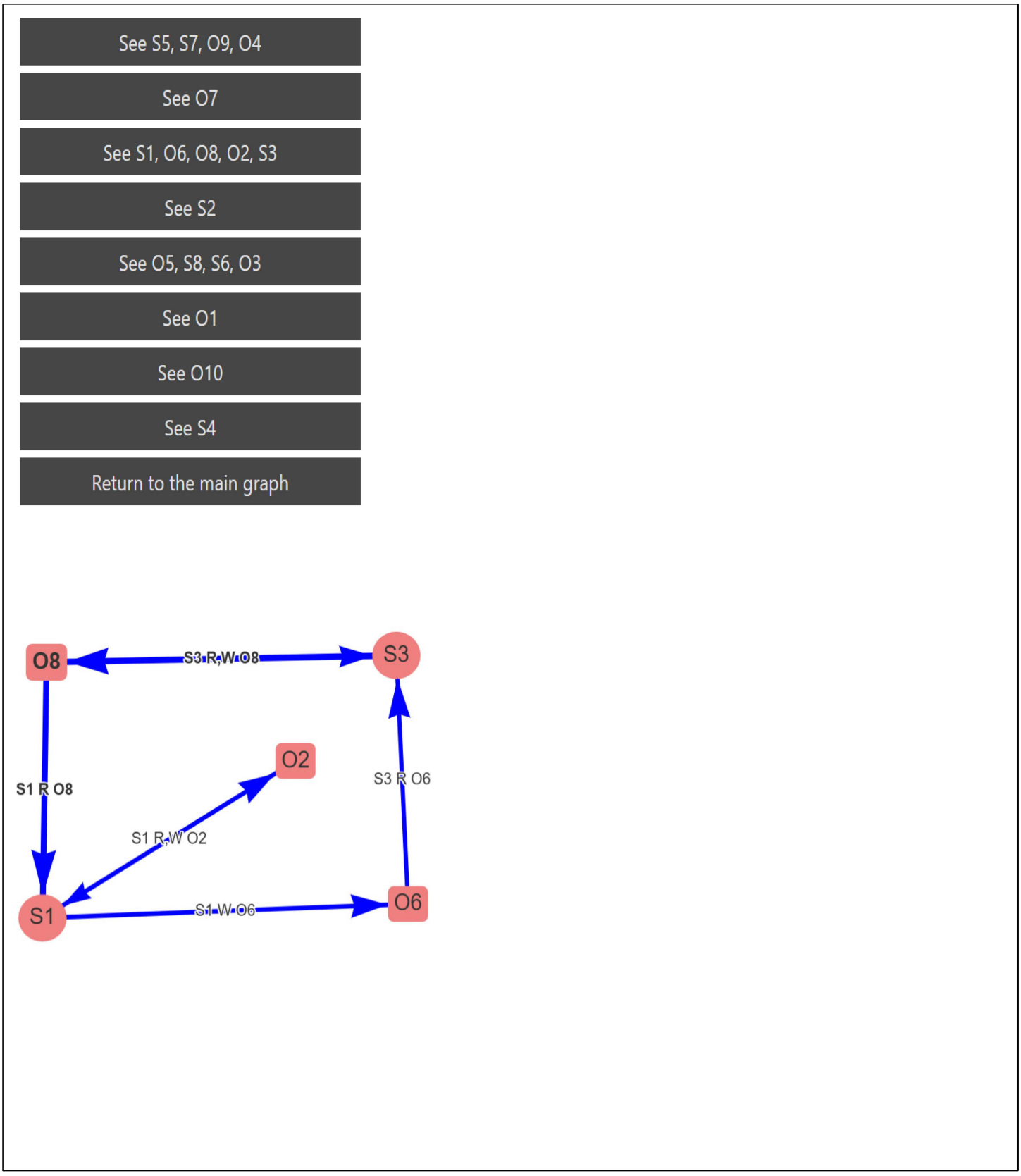


Figure 41 : Sous graphe (b), (Composant connexe {*S3*, *S1*, *O2*, *O6*, *O8*})

Suite du graphe

See S5, S7, O9, O4

See O7

See S1, O6, O8, O2, S3

See S2

See O5, S8, S6, O3

See O1

See O10

See S4

Return to the main graph

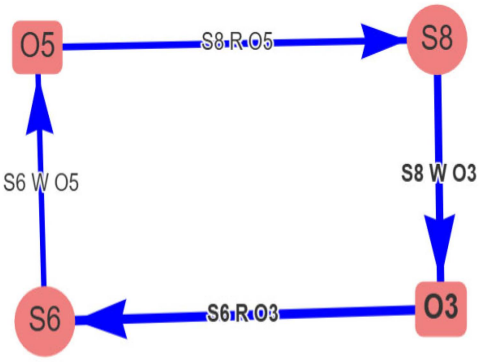


Figure 42: Sous graphe(c), (Composant connexe {*S6*, *S8*, *O3*, *O5*})

5.1.6. Simulation de temps d'exécution de l'algorithme

Pour évaluer la performance de l'algorithme, nous avons réalisés une simulation d'exécution du temps que l'algorithme peut prendre pour la détection des composantes fortement connexes (classes d'équivalence) ainsi que pour le calcul des étiquettes, pour un nombre élevé d'entités. Cette évaluation est réalisée avec une génération aléatoire d'entités et de matrices de contrôle d'accès et les résultats sont donnés dans un Graphe en courbe. Dans la Figure 43, nous avons utilisé 200 000 nœuds pour effectuer le test, et le temps de traitement a pris environ 54 secondes, ce qui montre que l'algorithme est très efficace et très raisonnable pour les systèmes à grande échelle. Ces essais nous a permis d'optimiser l'algorithme. L'intégration de ces optimisations dans le système assurera une évolutivité pour gérer des systèmes de grande taille tout en réduisant les temps de traitement. Nous avons suivi l'exemple de Stambouli et Logrippo [25]. Ces simulations sont faites pour matrices de contrôle d'accès de taille de plus en plus grande, jusqu'au point où notre ordinateur n'a été plus en mesure de compléter le calcul des étiquettes.

Étant donné que la performance en termes de temps d'exécution dépend de la capacité de l'ordinateur, nous avons utilisé un ordinateur i7 de 13e génération, 16 Go de RAM et 1 To de disque dur.

Nodes	Time (s)
100	0.000
500	0.001
1000	0.003
2000	0.006
5000	0.023
8000	0.049
10000	0.184
20000	0.392
50000	2.325
100000	11.761
120000	16.484
200000	54.599

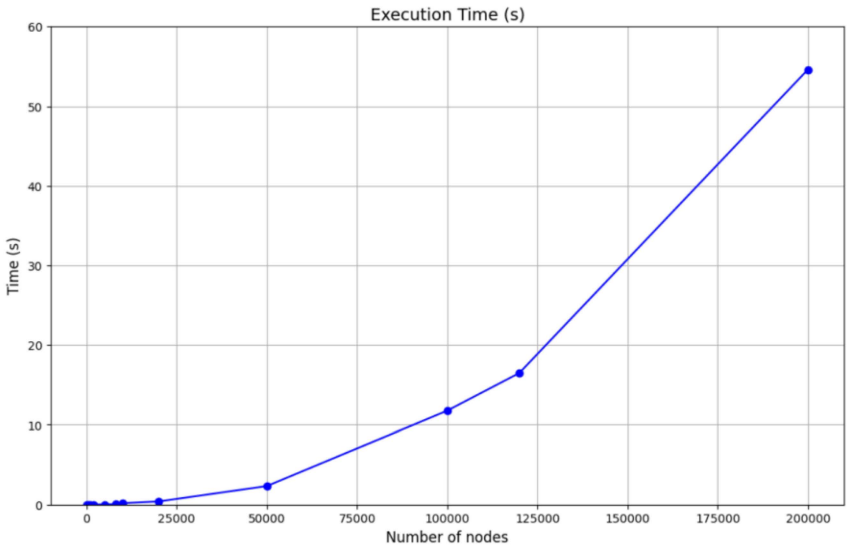


Figure 43: Temps d'exécution de l'algorithme

5.2. CONCLUSION

Le chapitre consacré à la réalisation du système a permis de présenter les étapes pratiques de développement et de mise en œuvre des fonctionnalités décrites lors de la conception. Il a détaillé l'intégration des différents modules, notamment l'application des modèles DAC, MAC, RBAC et ABAC, la gestion dynamique des permissions, ainsi que la génération des visualisations interactives.

La réalisation s'est appuyée sur des technologies adaptées, assurant la modularité, la performance et la convivialité attendues.

Les interfaces de saisie (importation de fichiers Excel et terminal de commande) ont été développées pour offrir une grande flexibilité d'utilisation.

Enfin, l'ensemble des fonctionnalités a été validé par des scénarios de test démontrant la conformité du système aux besoins fonctionnels et non fonctionnels spécifiés.

Le chapitre a également détaillé l'évaluation des performances de l'algorithme de propagation et de détection des composantes fortement connexes, démontrant sa capacité à traiter de grands volumes de données avec un temps de réponse maîtrisé.

L'ensemble de ces réalisations constitue une base solide pour les phases de validation et pour de futurs enrichissements fonctionnels.

CHAPITRE 6 : CONCLUSION GÉNÉRALE

Ce travail de recherche a permis de concevoir et de développer un système graphique interactif de gestion et de visualisation des politiques de contrôle d'accès basé sur des modèles formels, notamment le DAC, MAC RBAC et ABAC. À travers six chapitres, nous avons construit une réflexion complète et une mise en œuvre pratique de différentes approches de contrôle d'accès, en nous appuyant sur des modèles de gestion éprouvés.

Dans le premier chapitre, nous avons introduit le contexte de ce projet en raison de l'importance de la gestion des autorisations dans des systèmes où les interactions entre utilisateurs et ressources sont nombreuses et dynamiques. Les systèmes modernes, tels que l'IoT, nécessitent des modèles d'accès robustes et adaptés aux exigences spécifiques de sécurité et de confidentialité. Nous avons également défini les problématiques et les défis rencontrés, en soulignant la nécessité d'une gestion centralisée et flexible des autorisations.

Dans le deuxième chapitre, une revue de la littérature a été présentée, en analysant les différents modèles d'accès existants, y compris le modèle Discrétionnaire d'Accès (DAC), les modèles multi-niveaux (MAC), le modèle basé sur les rôles (RBAC) et le modèle basé sur les attributs (ABAC). Nous avons montré que chaque modèle présente des forces et des limitations selon le contexte d'application. Toutefois, l'intégration de plusieurs modèles permet de répondre de manière plus globale aux exigences des systèmes IoT. La revue a mis en évidence la flexibilité et la centralisation du modèle RBAC tout en explorant ses extensions pour intégrer des approches plus granulaires telles que l'ABAC et les mécanismes de contrôle de flux comme Bell-LaPadula.

Dans le troisième chapitre, nous avons décrit la théorie que nous avons utilisée pour la réalisation de notre système.

Le quatrième chapitre présente la conception détaillée du système de gestion des permissions et de contrôle des flux d'information. L'architecture générale a été présentée, affichant comment nous avons intégré plusieurs modèles de contrôle d'accès (DAC, MAC RBAC et ABAC) pour offrir une solution complète et modulaire. Nous avons également expliqué comment les composants du système interagissent entre eux, en soulignant les avantages de la visualisation graphique des relations entités-permissions et de la gestion de l'ordre partiel des étiquettes de

sécurité. De plus, les choix technologiques que nous avons utilisés, tels que l'utilisation de NetworkX pour la gestion des graphes et PyVis pour la visualisation, ainsi que l'algorithme de Tarjan pour gérer les composants fortement connexes.

La modélisation UML est utilisée pour représenter l'ensemble du système : un diagramme de cas d'utilisation identifie les principales interactions entre l'utilisateur et le système, un diagramme de classes structure les entités et leurs relations, un diagramme de séquence illustre l'enchaînement des messages lors des scénarios typiques, et un diagramme d'architecture montre la répartition logique des composants.

Le cinquième chapitre décrit la réalisation de notre système. Nous avons décrit les options d'exécution disponibles, à savoir l'option du fichier Excel et l'option des commandes manuelles, chacune étant adaptée à différents besoins de gestion. L'intégration de l'algorithme de Tarjan pour l'identification des composantes fortement connexes s'est avérée particulièrement efficace, tant sur le plan théorique que pratique. Les tests de performance menés sur des graphes allant jusqu'à 200 000 entités ont démontré une excellente scalabilité, avec un temps d'exécution inférieur à 1 minute. Ces résultats confirment que le système est capable de supporter des environnements à grande échelle, tels que ceux rencontrés dans l'Internet des Objets (IoT) ou les infrastructures critiques.

Enfin, le chapitre six donne la conclusion générale.

En somme, cette plateforme constitue une contribution significative à la compréhension, à la simulation et à l'évaluation des politiques de sécurité complexes, tout en offrant une base solide pour de futures extensions vers l'ABAC ou des mécanismes de détection d'anomalies.

Notre système constitue également un excellent outil d'expérimentation pour les étudiants. Il sera rendu disponible dans la Toile et il pourra être utilisé pour explorer et comprendre les divers modèles de contrôle de flux de données, ce qui renforce l'apprentissage et la recherche sur la théorie de sécurité du contrôle de flux des données.

On conclut par une récapitulation concise de nos contributions.

Ce travail de recherche et de développement avait pour ambition de répondre à plusieurs enjeux identifiés dès l'introduction :

- Concevoir un système de gestion et de visualisation des permissions qui soit à la fois modulaire, flexible et capable d'intégrer plusieurs modèles formels de sécurité.
- Permettre la simulation et l'expérimentation pédagogique des politiques de contrôle d'accès pour les environnements modernes tels que l'IoT.
- Proposer une solution interactive et performante, adaptée à la gestion de grands volumes de données.

Au terme de ce projet, les résultats obtenus attestent que ces objectifs ont été atteints :

- Nous avons conçu et développé une plateforme graphique interactive qui intègre les quatre modèles de contrôle d'accès annoncés (DAC, MAC, RBAC, ABAC), permettant de configurer et de comparer leurs mécanismes.
- L'architecture du système a été structurée en trois couches distinctes :
    - une couche d'entrée utilisateur (fichier Excel et terminal de commande),
    - une couche de traitement (application des règles et propagation des étiquettes),
    - et une couche de visualisation (graphes et tableaux).
- L'algorithme de Tarjan, choisi dès la phase de conception pour détecter les composantes fortement connexes et propager les étiquettes, a été implémenté et validé par des tests de performance confirmant sa scalabilité (traitement de graphes jusqu'à 200 000 entités en moins de 1 minute).
- Les promesses de transparence et de pédagogie ont été tenues grâce à la production de visualisations interactives et à l'intégration d'un module de suivi des violations de sécurité.
- La plateforme a été pensée comme un outil d'expérimentation pour les étudiants et chercheurs souhaitant explorer les approches de contrôle de flux et comprendre les différences entre les modèles, Il sera rendu disponible dans la Toile.

Ces résultats confirment la cohérence entre les intentions affichées au départ et les livrables produits.

Enfin, plusieurs pistes de travaux futurs pourraient prolonger et enrichir cette contribution :

- Développer un module de détection d'anomalies comportementales, renforçant la dimension proactive de la sécurité.
- Intégrer des tableaux de bord web multi-utilisateurs pour permettre une administration collaborative et à distance.
- Étendre le système aux architectures distribuées (IoT massivement réparti, environnements cloud) en tenant compte des contraintes de synchronisation et de cohérence des permissions.
- Proposer des fonctionnalités de benchmark comparatif entre les différents modèles de contrôle d'accès en fonction de critères métiers spécifiques.

En somme, cette plateforme constitue un socle solide, à la fois pratique et théorique, pour la simulation, la compréhension et l'évaluation des politiques de sécurité dans des environnements à grande échelle.

Nous avons rendu notre système **publiquement accessible** sur GitHub, et le site a également été **déployé sur Replit**.

Voici le lien GitHub :
https://github.com/RipamaR/controle-acces-streamlit/

Et voici le lien du site Replit, où vous pouvez effectuer vos tests :
https://controle-acces-app-2w7mz3cmnkvsx4dkszyjkx.streamlit.app/#controle-d-acces-rbac-dac-china-wall

Les récapitulatifs des commandes que nous avons développées sont les suivants, en version française et anglaise :

COMMANDES	EXPLICATION COMMANDES	RÉFÉRENCES
	Système d'entités et canaux	
AddEnt E1	Permet de créer une entité E1 sans spécification de sujet ou d'objet.	Tableau 18, Page 61
AddCh E1 E2	Permet de créer un canal entre E1 et E2 sans distinction sujet/objet.	Tableau 19, Page 62
RemoveCh E1 E2	Permet de retirer le canal entre E1 et E2	Tableau 20, Page 62
	Commandes DAC	
AddSub S2	Permet de créer un sujet *S2*.	Tableau 11, Page 66
S2 AddObj O2	Création de l'objet O2 dont le propriétaire est S2.	Tableau 12, Page 73
S2 Grant S3 O2 R	Permission de lecture accordée à S3 par le propriétaire S2 sur O2.	Tableau 12, Page 73
	Commandes MAC	
AddSub S1	Permet de créer un sujet *S1*.	Tableau 11, Page 66
AddObj O1	Permet de créer un objet *O1*.	Tableau 11, Page 66
AddCh S1 R O1	*S1* a la permission de lecture sur *O1*.	Tableau 11, Page 66
RemoveSub S1	Supprime le sujet *S1*.	Tableau 11, Page 66
RemoveCh S1 R O1	Retrait de canal de lecture *S1* vers *O1*.	Tableau 11, Page 66
modifyCh S1 R O1 S1 W O1	Modifie la permission de *R* en *W* pour *S1* -> *O1*.	Tableau 11, Page 66
	Commandes Muraille de chine	
AddSub S1	Permet de créer un sujet *S1*.	Tableau 15, Page 83
AddObj O1	Permet de créer un objet *O1*.	Tableau 15, Page 83
Never {S1, O1}	Pour exprimer la contrainte que la combinaison S1, O1 ne peut paraître dans aucune étiquette.	Tableau 15, Page 83
Never {O1, O2} for {S3}	Pour exprimer la contrainte selon laquelle l'entité *S3* ne pourra jamais avoir des étiquettes contenant les combinaisons *O1, O2*.	Tableau 15, Page 83
AddCh S1 R O1	Lecture S1→O1 : viole la règle Never {S1, O1}.	Tableau 15, Page 83
	Commandes RBAC	
AddObj O1	Permet de créer un objet *O1*.	Tableau 16, Page 87
AddRole R1	Permet de créer un rôle *R1*.	Tableau 16, Page 87
GrantPermission R1 R O1	Attribue la permission *R* sur *O1* au rôle *R1*.	Tableau 16, Page 87
AddSub S1 R1	Crée un sujet *S1* avec le rôle *R1*.	Tableau 16, Page 87
RevokePermission R1 W O1	Retire la permission d'écriture du rôle *R1*	Tableau 16, Page 87
DeassignUser S1 R1	Retire le rôle du sujet *S1* (désassignassions).	Tableau 16, Page 87
ModifyPermission R1 R O1 W	Modifie la permission *R* sur l'objet *O1* accordée au rôle *R1*, pour qu'elle devienne *W*.	Tableau 16, Page 87
RemoveRole R1	Supprime le rôle *R1*.	Tableau 16, Page 87

Tableau 18: Liste des commandes implémentées (version française)

COMMANDS	COMMAND EXPLANATION	REFERENCES
	Entity and Channel System	
AddEnt E1	Creates entity **E1** without specifying subject or object.	Table 10, Page 61
AddCh E1 E2	Creates a channel between **E1** and **E2** without subject/object distinction.	Table 22, Page 62
RemoveCh E1 E2	Removes the channel between **E1** and **E2**.	Table 23, Page 62
	DAC Commands	
AddSub S2	Creates subject **S2**.	Table11, Page 66
S2 AddObj O2	Creates object **O2**, owned by subject **S2**.	Table 12, Page 73
S2 Grant S3 O2 R	Read permission on **O2** granted to **S3** by the owner **S2**.	Table 12, Page 73
	MAC Commands	
AddSub S1	Creates subject **S1**.	Table 11, Page 66
AddObj O1	Creates object **O1**.	Table 11, Page 66
AddCh S1 R O1	***S1** is allowed to read **O1**.*	Table 11, Page 66
RemoveSub S1	Deletes subject **S1**.	Table11, Page 66
RemoveCh S1 R O1	Removes the read channel from **S1** to **O1**.	Table 11, Page 66
modifyCh S1 R O1 S1 W O1	Modifies permission from **R** to **W** for **S1 → O1**.	Table 11, Page 66
	China Wall Commands	
AddSub S1	Creates subject **S1**.	Table 15, Page 83
AddObj O1	Creates object **O1**.	Table 15, Page 83
Never {S1, O1}	Expresses that the combination **S1–O1** can never appear in any label.	Table 15, Page 83
Never {O1, O2} for {S3}	Subject **S3** can never receive labels containing **O1–O2**.	Table 15, Page 83
AddCh S1 R O1	Read attempt **S1 → O1**, but violates the restriction **Never {S1, O1}**.	Table 15, Page 83
	RBAC Commands	
AddObj O1	Creates object **O1**.	Table 16, Page 87
AddRole R1	Creates role **R1**.	Table16, Page 87
GrantPermission R1 R O1	Grants **read** permission on **O1** to role **R1**.	Table 16, Page 87
AddSub S1 R1	Creates subject **S1** with role **R1**.	Table 16, Page 87
RevokePermission R1 W O1	Revokes **write** permission from role **R1**.	Table 16, Page 87
DeassignUser S1 R1	Removes role **R1** from subject **S1**.	Table 16, Page 87
ModifyPermission R1 R O1 W	Modifies the permission of **R1** on **O1** from **R** to **W**.	Table 16, Page 87
RemoveRole R1	Deletes role **R1**.	Table 16, Page 87

Tableau 19:Liste des commandes implémentées (version anglaise)

❖ Justification des outils graphiques non choisis

Au cours du développement du système de visualisation des flux de données, plusieurs bibliothèques et outils graphiques ont été examinés afin d'identifier la solution la plus adaptée aux besoins du projet.

Certains de ces outils n'ont pas été retenus pour des raisons spécifiques, que nous présentons brièvement dans le tableau suivant :

OUTIL EXAMINE	DESCRIPTION	AVANTAGES OBSERVES	LIMITES CONSTATEES	RAISONS DU REJET
PLOTLY / DASH	Framework web interactif basé sur Plotly.js.	Interface web moderne et interactive.	Nécessite un serveur Dash complet et une structure plus lourde.	Non retenue pour éviter une complexité de déploiement inutile.
GRAPHVIZ	Générateur de graphes orienté structures hiérarchiques.	Excellente lisibilité structurelle, export PDF/SVG.	Pas d'interactivité, ajustement manuel du positionnement.	Utilisée ponctuellement pour les diagrammes UML, mais pas pour le rendu dynamique.
BOKEH	Bibliothèque Python pour visualisations web.	Support du zoom et de l'interaction via navigateur.	Testé sur plusieurs scénarios d'erreurs.	Moins flexible pour les graphes orientés complexes.

Tableau 20: Outils graphiques examinés

RÉFÉRENCES

[1] D. F. Brewer and M. J. Nash, "The Chinese Wall Security Policy," in *S&P*, 1989, pp. 206-214.

[2] M. Alramadhan and K. Sha, "An overview of access control mechanisms for internet of things," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017: IEEE, pp. 1-6.

[3] B. W. Lampson, "Protection," *ACM SIGOPS Operating Systems Review,* vol. 8, no. 1, pp. 18-24, 1974.

[4] D. E. Bell and L. J. La Padula, "Secure computer system: Unified exposition and multics interpretation," 1976.

[5] D. F. Ferraiolo, R. Kuhn, and R. Chandramouli, "Role-Based Access Control, 2nd edn. Artech House," *Inc., Norwood,* vol. 25, 2007.

[6] 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.

[7] R. Chen and J.-J. Lévy, "Une preuve formelle de l'algorithme de Tarjan-1972 pour trouver les composantes fortement connexes dans un graphe," in *JFLA 2017-Vingt-huitièmes Journées Francophones des Langages Applicatifs*, 2017.

[8] E. Bertin, D. Hussein, C. Sengul, and V. Frey, "Access control in the Internet of Things: a survey of existing approaches and open research questions," *Annals of telecommunications,* vol. 74, pp. 375-388, 2019.

[9] H.-C. Chen, "Collaboration IoT-based RBAC with trust evaluation algorithm model for massive IoT integrated application," *Mobile Networks and Applications,* vol. 24, no. 3, pp. 839-852, 2019.

[10] B. Lampson, "" Protection", In 5th Princeton Symposium on Information Science and Systems," *March1971,* pp. 437-443, 1971.

[11] G. S. Graham and P. J. Denning, "Protection: principles and practice," in *Proceedings of the May 16-18, 1972, spring joint computer conference*, 1971, pp. 417-429.

[12] B. Lampson, "Protection and access control in operating systems," *Operating Systems, Infotech State of the Art Report,* vol. 14, pp. 309-326, 1972.

[13] B. Maboudou, "Un environnement pour la modélisation des systèmes de contrôle d'accès," Université du Québec en Outaouais, 2015.

[14] K. J. Biba, "Integrity considerations for secure computer systems," 1977.

[15] L. Logrippo, "Mandatory Access Control Contrôle d'accès obligatoire à niveaux (Méthodes de contrôle de flux) ", 2013. Accessed: 31/10/2024. [Online]. Available: http://w3.uqo.ca/luigi/INF6153ContrAcc/20MAC.pptx

[16] R. S. Sandhu, "Lattice-based access control models," *Computer,* vol. 26, no. 11, pp. 9-19, 1993.

[17] A. Capozucca, M. Cristiá, R. Horne, and R. Katz, "Brewer-Nash Scrutinised: Mechanised Checking of Policies featuring Write Revocation," *arXiv preprint arXiv:2405.12187,* 2024.

[18] D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (RBAC): Features and motivations," in *Proceedings of 11th annual computer security application conference*, 1995, pp. 241-48.

[19] R. Cao, "Research on RBAC Based Role Access Control in Financial MIS," in *2022 6th International Conference on Wireless Communications and Applications (ICWCAPP)*, 2022: IEEE, pp. 147-150.

[20] N. Benaïssa, "La composition des protocoles de sécurité avec la méthode B événementielle.(Security protocols composition using Event B)," Henri Poincaré University, Nancy, France, 2010.
[21] V. C. Hu *et al.*, "Guide to attribute based access control (abac) definition and considerations (draft)," *NIST special publication,* vol. 800, no. 162, pp. 1-54, 2013.
[22] L. Logrippo, "The order-theoretical foundation for data flow security," *arXiv preprint arXiv:2403.07226,* 2024.
[23] L. Logrippo, "Data flow security in Role-Based Access Control." [Online]. Available: https://www.site.uottawa.ca/~luigi/papers/24_RBAC-FLOW.pdf
[24] L. Logrippo, "Multi-level access control, directed graphs and partial orders in flow control for data secrecy and privacy," in *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10*, 2018: Springer, pp. 111-123.
[25] A. Stambouli and L. Logrippo, "Data flow analysis from capability lists, with application to RBAC," *Information Processing Letters,* vol. 141, pp. 30-40, 2019.
[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.
[27] M. Ben-Or, "Lower bounds for algebraic computation trees," in *Proceedings of the fifteenth Annual ACM Symposium on Theory of Computing*, 1983, pp. 80-86.
[28] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM Journal on Computing,* vol. 1, no. 2, pp. 131-137, 1972.
[29] R. BS, N. K. NV, and R. Shyamasundar, "Towards unifying RBAC with information flow control," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021, pp. 45-54.
[30] S. Chakraborty, R. Sandhu, and R. Krishnan, "On the feasibility of attribute-based access control policy mining," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 2019: IEEE, pp. 245-252.
[31] M. Bishop, *Computer security, Art and science*. Addison-Wesley, 2019.
[32] R. S. Sandhu, "Lattice-based enforcement of chinese walls," *Computers & Security,* vol. 11, no. 8, pp. 753-763, 1992.